NORTHERN BYTES

Volume 5 Number 4

Welcome to another chapter in the continuing saga of, "Can a small newsletter devoted to the TRS-80 Models I, III, and 4 survive in a world of catalog-size publications?"

I'm writing these lines in Mid-March, and up here that means we still have about a foot of snow on the ground. A couple of nights ago it got down to 20 below (Fahrenheit), today we're having a heat wave - it's all of 40 degrees! I mention this to point out that we now have something in common with those "big" publications - it's called "lead time". And what that means to you is that if you type in a program from NORTHERN BYTES, you should be sure to get at least the next two or three issues to make sure you get any corrections that might appear. You folks have been pretty good about letting us know when you find a bug in one of the programs we publish, and believe me, we do appreciate it!

Most of the program listings in NORTHERN BYTES will eventually find their way onto a TAS Public Domain Library disk. If you get an early version of the disk, you might get a "buggy" program, so when you see a correction in NORTHERN BYTES, you might check your PD disks to see if the corrections have been applied. One caution - we sometimes renumber assembly language source code listings on the disks after making corrections, so you should probably first compare the listing on the disk to the original listing in NORTHERN BYTES, and THEN check to see if corrections have been applied. When we do make a correction on one of the Public Domain disk programs, we'll try to remember to include a comment line to that effect at the start of the program.

Perhaps I'm getting ahead of myself a bit. Some of you may not be familiar with the TAS Public Domain Software Library. If that's the case, you should check it out. For $10, you get a "flippy" double-sided disk (works in all standard 40-track disk drives), crammed full of Public Domain programs. As I write this, we have two such disks, perhaps there will be more by the time you get this issue. We are aiming for QUALITY, not QUANTITY, in this library, so I doubt that you will ever see hundreds of disks in the library. If you have some good public domain programs you'd like to put in the library, by all means send them in (but PLEASE don't feel obligated to "stuff" the disk you send - we would MUCH rather have a disk with ONE or TWO good quality Public Domain programs, with documentation, than every program you've ever downloaded from your local Bulletin Board System)! And, if you'd like a table of contents for the most recent disks in the library, send a self addressed, stamped envelope to The Alternate Source, 704 North Pennsylvania Avenue, Lansing, Michigan 48906.

Turning to other things, I'm gratified by the number of article submissions I've received recently. Considering that we aren't paying for articles (except to put you on the mailing list to automatically receive the next six issues of NORTHERN BYTES, if we publish that many more), I'd say that some of you folks are mighty generous, and we thank you for that. Folks, I'd encourage you to submit articles to NORTHERN BYTES as a way to keep a dialogue going among TRS-80 users. When the rest of the world seems to be tilting between IBM-compatibles, computers with "fruity" names, and low-cost game machines, we know we have a good, solid computer that has stood the test of time. The Tandy line has something for everyone, but a lot of software and hardware manufacturers have jumped onto the IBM bandwagon. That's okay, let them slug it out with the big boys if they want to - we know that there's still many hundreds of thousands of TRS-80 users out there! Maybe now we can all learn from each other.

You know, it's a funny thing. The Apple people started out handing out technical information on the Apple to all comers. Their manuals documented all the PEEKs and POKEs you might ever want to use. They encouraged "outside" support and user groups, and the Apple took off. Tandy took the opposite approach, providing as little technical information as possible, discouraging outside vendors from selling TRS-80 compatible products in any way possible, and refusing to aid or even acknowledge the existence of user groups. How times have changed! The Model 4 Technical Reference Manual is a masterpiece, a truly useful documentation of both the hardware and software in the Model 4. Tandy now is more tolerant of outside support, they even encourage it where it doesn't directly compete with existing Tandy products. The TRS-80 Microcomputer News now carries information on new computer clubs and user groups. And, what of Apple? From what I've heard, they're going the other way. The Macintosh is a BUSINESS machine, thank you, and Apple Computer will tell you what you need to know about it. PEEKs and POKEs? Assembly language calls? Technical Information? Why would you want that, unless you're a (shudder) "hacker" - and "hackers" don't buy BUSINESS machines...

In the middle of all the commotion, Tandy is quietly selling computers, and most TRS-80 owners wouldn't part with their machines for anything else in a comparable price range (except, perhaps, for those who are writing commercial software and fear that they are going to be "left behind"). I've also heard tales of folks who are using the Tandy hi-resolution modification, or similar hi-res hardware, and developing "windowing" type software. I suspect that as we all continue to learn from each other, we may find that the Tandy computer line isn't dead yet - and we may see innovations in software we never dreamed possible, right on our old Model I/III/4 computers!

We encourage you to keep in touch with us. Send us your articles, programs, patches, hints, tips, techniques, or anything else you'd like to share with the community of TRS-80 users as a whole. Keep in mind that we can receive electronic mail through the MCI Mail system, the MCI Mail address is 102-7413. Overseas readers with access to TELEX service can send us a TELEX message at 6501027413 (answerback is 6501027413 MCI). If you have a choice of area codes for the U.S.A., use the one for Western Union or MCI International TELEX numbers. When you send us MCI Mail or TELEX messages, we can download them right into a TRS-80, and use the search-and-replace capabilities of a word processing program to remove the extra carriage returns. That makes it easy for us, because we don't have to re-type your material! By the way, MCI Mail has a new nationwide toll-free number that supersedes the one published in Volume 5, Number 2 of NORTHERN BYTES - it's (800) 323-0905. If that one is busy, the number we published should still work, it's (800) 323-7751.

One final note - I'm really surprised that no one has taken us up on our offer to run non-commercial "unclassified" ads for 10 cents per word per issue (we'll even throw in your name, address, and phone number for free). Payment must be made in advance, or you may charge it to your VISA or MASTERCARD. TRS-80 user groups may run a FREE ad of up to 25 words (NOT counting name, address, and phone number) for any purpose except to sell products (free user group ads will run once, but you may re-submit them as often as you like). Display advertising is also available, for information on that contact Charley Butler at The Alternate Source (phone (517) 482-8270).

I hope you enjoy this issue of NORTHERN BYTES!

THE EXTERMINATOR - Flyswatters ready, everyone? Time to kill the BUGS that have appeared in previous issues of NORTHERN BYTES!

The SETDATE/CMD program featured in NORTHERN BYTES Volume 5, Number 2 (and as found on early versions of TAS Public Domain Library Disk #001) had a couple of problems that would affect some users but not others. Michael Davis found a bug that affects DOSPLUS 3.5 users that have the KI/DVR program installed. The problem is that at line 2040 the program makes a CALL to 2BH, the keyboard scan routine. The following instruction (line 2050) is JR Z,GETKEY, which is intended to jump if no key was pressed. According to TRS-80 ROM ROUTINES DOCUMENTED (whose author shall remain nameless at this point), on exit from the routine at 2BH, the Z flag will be set and the A register will contain zero if no key was pressed. Well, 'taint so. The Z flag is NOT necessarily set if no key was pressed, although

the A register DOES contain zero. So, to fix this bug, add the following line to the SETDATE/CMD source code:

```
2045    OR      A       ;Z flag set if no key
```

By the way, this bug won't affect many users, except to make the keyboard response a bit sluggish. However, when the DOSPLUS KI/DVR program is used, SETDATE/CMD refuses to respond to any keystrokes - it just locks up!

That's not the end of the problems with SETDATE/CMD. I knew that the fact that I didn't CLOSE an open file would bother some folks. I've used the program on several DOSes as is, with no problems. But, for those of you that just can't stand the thought of not closing a file once it has been opened, you have a friend in Paul Snively. He not only sides with you, he provides a solution. Here are Paul's comments:

"I couldn't help but think that not closing SETDATE/CMD was a little kludgey, and even dangerous (a phobia, no doubt, caused by Model I TRSDOS, which has a nasty habit of trashing out the directory if you're fool enough to KILL an open file!!). Anyway, the problem, of course, is that when you call 443FH (to REWIND to record 0 of the file,) the DOS apparently insists on resetting the EOF value to record 0! Makes sense, right? However, there is something you can do to avoid that!

"What we need to do is put the DOS into 'direct access mode'. This is a nasty piece of jargon which basically means 'don't change the EOF, no matter what happens!' To make your program go into direct access mode, add the following lines to SETDATE/ASM:

(Note: This will NOT work with TRSDOS 1.31)

```
2171    INC     DE      ;Point to FCB+1
2172    LD      A,(DE)  ;Get status byte from FCB
2173    OR      40H     ;Set bit 6
2174    LD      (DE),A  ;Re-save status byte
2175    DEC     DE      ;Point to FCB
```

"So far so good. This, of course, means that you can now close the file without fear. Get rid of the warning about closing the file (lines 2360 and 2370), and instead, make line 2360 read:

```
2360    CALL    4428H   ;DOS CLOSE routine
```

which will, of course, close the file. Viola! One more kludge bites the dust! You might want to keep bit 6 of relative byte 1 of the FCB in mind when developing future applications that have occasion to write to someplace in the middle of the file."

Shortly after receiving Paul's letter, I had a chance to speak with him by phone, and he more or less acknowledged that users of the SETDATE/CMD program would probably not experience any difficulty with the code as written as long as it was used with any of the more intelligent DOSes (DOSPLUS, MULTIDOS, NEWDOS/80 et al.). However, the Model I version of TRSDOS (version 2.3 and earlier) had a definite problem in that under certain circumstances, if you OPENed a file, then did not properly CLOSE it and then KILLed it, you could write garbage all over the directory! I do not know if this could happen with SETDATE/CMD running under TRSDOS 2.3, but I wouldn't want to find out the hard way. If anyone is currently using SETDATE/CMD with TRSDOS 2.3, you might want to write in and let us know if you've had any problems.

One final comment on SETDATE/CMD! Dexter Walker wrote in to tell me that the patch that I published to disable the DATE and TIME prompts in TRSDOS 1.3 didn't work properly for him. He then tried Bob Snapp's DOSFIX05 patch, which DID work. So, if you tried the patch I published and had problems, try Bob Snapp's patch and see if that works. Here's the patch:

PATCH #0 (ADD=4EB8,FIND=213B51,CHG=C3394F)

That concludes the comments about SETDATE/CMD, but there was another bug in that same issue of NORTHERN BYTES, and Paul Snively gets the eagle-eye award - he found it! His letter continues as follows:

"While I'm here, let me point out a slight misunderstanding in Northern Bytes Volume 5 number 2. The 'Model III TRSDOS 1.3 bug' that was downloaded from Compuserve and written by Morris 'Mojo' Jones is in error. The fact that TRSDOS 1.3 takes the LRL from the directory entry on an open instead of from the B register is NOT A BUG! It is a feature of TRSDOS 1.3 that was stolen from guess who? Vern Hester! Yup, MULTIDOS has that same 'bug.' Why? That's a tricky one. It basically has to do with the occasional oddball who writes a program which actually uses the LRL. A good case in point is SuperScripsit. If you look at the Model III version of SuperScripsit (which is, of course, distributed on TRSDOS 1.3) you will find that the directory entry for ERRORS/CTL (the error message file) has a LRL of 40H. Let's see, 40H... Why, that's 64 in decimal! Hmmm... Why the weird LRL? You guessed it - error messages in SuperScripsit are characters long. Why do the error messages this way, though?

"Well, it's so 'they' (whoever wrote SuperScripsit) could cram four (256 / 64) error messages into a single disk sector and let the DOS (with its LRL capabilities) figure out which error message needed to be accessed in order to retrieve a particular error message. In other words, with proper use of the LRL byte, you can let the DOS worry about how to use disk space most efficiently. Got that? Good.

"Now look at the Model I version of SuperScripsit. Notice that the LRL for ERRORS/CTL is 0, which means 256! Yup, Model I SuperScripsit uses the LRL to figure out which error message to yank, BUT IT PASSES THE LRL TO THE OPEN ROUTINE VIA THE B REGISTER!

"You knew that, right? Most DOSes use the B register to pass the LRL to the OPEN vector. However, TRSDOS 1.3 gets it from the directory entry. SO DOES MULTIDOS! (And MULTIDOS did it first!!) That's why Vern has published a zap to the directory entry of ERRORS/CTL on the Model I so that the error messages returned by SuperScripsit running under MULTIDOS on the Model I are the right length! In other words, MULTIDOS and TRSDOS 1.3 handle the LRL the same way, and it's not a bug, it's a means of getting the LRL right... 'right,' in this case, meaning making it the same as it was when the file was written. Understand? Good. If you have any questions about this very important difference between those two DOSes (MULTIDOS and TRSDOS 1.3) and the rest of the world, ask me (Paul Snively, Ashton-Hershey 12, Indiana University, Bloomington, Indiana 47405 - telephone (812) 337-1570), or better yet, ask Vern. After all, it was his idea."

Moving up to the last issue of NORTHERN BYTES (volume 5 number 3), Nathan W. Harrington had a few comments about creating your own NEWDOS/80 /SYS overlays that weren't covered in the article. Nathan offers these comments:

"First of all, about the byte you load into the A register. It is actually 3 individual numbers, not 2 as you mentioned. The by is divided as UUU BB SSS. SSS + 2 = the relative sector in the directory containing the (F)ile (D)irectoy (E)ntry (FDE). BB * 32 (20H) = the offset in the sector to the FDE. UUU = a user defined code greater than zero.

"As you know, the standard directory has ten sectors (numbered 0-9) [NEWDOS/80 users have the option of specifying an enlarged directory by specifying a FDRIVE DDGA parameter of greater than two, which will expand the number of disk sectors allocated to the directory, but the additional sectors are not used for /SYS overlay filenames. Therefore, we need only concern ourselves with sectors 0-9, regardless of the actual size of the directory -ed]. The first sector, sector 0, is reserved for the (G)ranule (A)llocation (T)able (GAT) and some system information such as diskette name, password, date, and the AUTO command used at boot up. Sector 1 is reserved entirely for the (H)ash (I)ndex (T)able (HIT). This leaves eight sectors (sectors 2-9) for directory entries (FDE's). Each sector has 256 bytes which are divided into eight 32-byte FDE's. 8 * 8 = 64 less one for BOOT/SYS and one for DIR/SYS, leaves the 62 FDE's you see in a directory of a formatted [data] diskette containing no other files [assuming that the FDRIVE DDGA was set to the standard value of two when the diskette was formatted -ed].

"All SYSTEM files (BASIC/CMD is <u>not</u> a SYSTEM file) must reside in the first four FDE's of each of the eight remaining directory sectors. These include BOOT/SYS and DIR/SYS. 8 sectors * 4 FDE's/sector for SYSTEM use = 32 FDE's for SYSTEM use - SYS0/SYS through SYS29/SYS (30 files), BOOT/SYS, and DIR/SYS = 32.

"These are organized in a very orderly fashion. Assume that the SYSTEM files are in one big list starting with BOOT/SYS and DIR/SYS, followed by SYS0/SYS - SYS29/SYS in numeric order. Then, you start filling the directory sectors as follows. The FDE's will be referenced by their first byte (the relative byte in the sector), i.e. they are numbered 00, 20, 40, 60, 80, C0, and E0. Note that these are HEXADECIMAL numbers and that 20H = 32, which is how many bytes there are per FDE.

"First, you will take the file names directly from the list -- IN ORDER. Fill all FDE's numbered 00 in each of the eight sectors [the eight sectors numbered 2-9, and don't forget that

BOOT/SYS and DIR/SYS go into the first two slots, followed by SYS0/SYS, SYS1/SYS, SYS2/SYS and so on... -ed]. Then, go back to sector 2 and begin filling the FDE's numbered 20. Continue this method until all files are in the directory. If you would include SYS22/SYS - SYS29/SYS, you would end up in sector 9 at FDE 60. This brings us to the BB portion of the byte loaded into the A register. BB * 32 is the relative byte within the directory sector of the beginning of the FDE associated with the overlay. Since BB is only two bits, the possible values are 00, 01, 10, 11 (binary) which gives [corresponds to] 00, 20, 40, and 60 HEXADECIMAL. Therefore, all SYSTEM files must also reside in the directory in one of the first four FDE's of each sector.

"The SSS portion of the byte is, as you stated, the number of the directory sector plus two.

"The UUU portion of the byte is a user defined code. In the case of the current SYSTEM files (SYS0/SYS - SYS21/SYS), it is a NEWDOS/80 defined code. This code can make it possible to have more than one routine in the overlay. By setting various combinations of bits, you can arrive at seven different codes. 000 is not used (nor available for use) so you can have 001, 010, 011, 100, 101, 110, and 111 (binary). It should be noted also that NEWDOS/80 uses the C register in many cases to obtain an even greater number of routines. By loading A and C, it is theoretically possible to have 7 (in A) * 256 (in C) = 1792 different routines.

"As an example, 3FH would be 001 11 111 which would be routine number 1, sector 9, offset 60H (3 * 20H). If the format is followed correctly, this would be SYS29/SYS.

"You may call other DOS routines in this manner also. I have started a list of the values needed to do this, but it is incomplete. The B register is also used for some purpose which is as yet unclear to me. If you look in SYS1/SYS, you will find all the commands, each followed by three bytes. These bytes, or part of them (a portion of each byte, that is), are loaded into the A, B, and C registers. The overlay then decodes this value and runs the appropriate routine.

"You might also mention that people should start with SYS29/SYS and work down toward SYS21/SYS as Apparat has said that if they make any additions, they will start with SYS22/SYS.

"I hope this information is of use to you in making the procedure more clear. I might also refer you to the NEWDOS/80 manual chapter five about the directory, and the last page of chapter twelve."

Thanks to Nathan for the additional information. I knew about the UUU BB SSS format but chose to express it differently in order to (hopefully) make it a bit easier for the average reader to understand. I suspect that most readers will understand the above concepts more easily if they actually look at the directory of a NEWDOS/80 system disk (using SUPERZAP), and note the pattern of the /SYS overlay filenames as they are stored in the directory.

If you missed the original article but have followed the above discussion so far, you might be interested to know that once the A register is properly loaded, you simply execute an RST 28H instruction to call the overlay (the /SYS overlay filename must be in the correct directory slot!). The contents of the A and C registers are preserved (as Nathan mentioned above), so these can be tested by the initial code of the overlay to determine which part of the overlay is to actually be executed.

TRS-80 ROM ROUTINES DOCUMENTED BUG - As you may be aware, I have been documenting the bugs that have appeared in "TRS-80 ROM Routines Documented", so that those of you with early editions of the book can make corrections. As mentioned earlier, the routine at 002BH (documented on page 12) does NOT necessarily set the Z flag when the A register contains zero (indicating no key pressed), contrary to what the book says. In addition, there's a simple typo to report! Near the top of page 49 it says, "PROGRAMMING HINT: As with the previous routine (at 28B7H)..." Trouble is, the "previous routine" was actually at 2857H. This also throws off the "Hexadecimal Address Cross-Reference" on page 124 - delete the reference to 28B7 there, and add page 49 to the reference to 2857. If you purchased your copy of the book recently, these corrections may have already been made. If you have an older copy, and have made the above correction plus the corrections that have appeared in previous issues of Northern Bytes, your copy of "TRS-80 ROM Routines Documented" should be up to date!

The 002BH bug was discovered by Michael Davis, and Paul Snively found the typo (I'm thinking about appointing Paul as official proofreader!). Paul also notes that on page 48, I document the routine at 28BFH with the notation that it will "make room for a string in the string storage area if possible (will go to an Out of String space Error if not possible)." Paul thinks that I probably should have pointed out that this is Microsoft's infamous "garbage collection" routine that has plagued so many users, even though it's fairly obvious from the above description. So, in case you didn't catch the implication of this routine the first time, you now know where all the garbage is collected!

TRS-80 TIDBITS, TRASH, TREASURE, AND TRIVIA by John Hallgren, 1939 Atlantis Drive, Clearwater, Florida 33515 (telephone (813) 797-5125 8-11 P.M. Eastern Time and weekends). This article is excerpted from John's series in ELSE.

[The following item is from the November/December 1983 issue of ELSE!]

* This item should have appeared about two years ago when the Model III was fairly new, but the conditions which caused me to develop this patch only occured last week. What do you do if you're at a remote site, trying to move a file from a Model I single density disk to a Model III TRSDOS 1.3 disk, and the file you need is the last one that the CONVERT program is processing and you get a DISK SPACE FULL error? And to compound the problem, you don't have any blank ones with you and nobody else will let you borrow one? Well, I just gave up, went home and started taking apart the CONVERT program from TRSDOS to see if another query prompt could be added.

I wanted it to display the next filename to be converted and prompt for a Yes/No/Quit response. The program already had the code to display the prompt (Y/N/Q) and process the response when the filename already exists on the destination disk, so all I had to do was to interface to that routine and in the smallest number of bytes. Before these patches can be done, the patch to disable password checking must have been applied. For those who missed it the first two times:

PATCH *2 (ADD=4ED4,FIND=20,CHG=18)

Now that we have made sure we can modify any file, here's the patches:

```
PATCH CONVERT/CMD (ADD=520A,FIND=06,CHG=1C)
PATCH CONVERT/CMD (ADD=5511,FIND=21955B,CHG=CD065A)
PATCH CONVERT/CMD (ADD=5536,FIND=ED53,CHG=135A)
PATCH CONVERT/CMD
        (ADD=5A06,FIND=4D6F64656C20,CHG=3E86CD155A21)
PATCH CONVERT/CMD
        (ADD=5A0C,FIND=3120746F204D,CHG=955BC0E1C39E)
PATCH CONVERT/CMD
         (ADD=5A12,FIND=6F64656C2033,CHG=553E7032F153)
PATCH CONVERT/CMD
        (ADD=5A18,FIND=20436F6E7665,CHG=CDED53C9436F)
PATCH CONVERT/CMD
              (ADD=5A1E,FIND=7273696F6E,CHG=6E76657274)
```

I don't know what's worse, coding and testing the changes or converting them to TRSDOS patch format! As always, I hope this might be of use to somebody. Now, what else can I do to TRSDOS??

[The following items are from the July/August 1983 issue of ELSE!]

0. -- Warning ! This article will deal only with the Model I Radio Shack Double-Density kit. If you're not interested, go on to the next article please. Anybody still here? Well, I guess it's just you and me kid. As I stated in the last article, one of the reasons I bought the R/S doubler was to hopefully assist other owners who may have less technical software knowledge. I have determined how most of TRSDOS 2.7DD works, and it's basically the Model III TRSDOS 1.3 with some enhancements, a few bugs and the usual minor quirks. Now for the details.

1. -- My R/S Shugart disk drive works just fine (in a 35 track, 40 ms mode) with the doubler. Some people told me that it would have difficulty formatting, writing and reading in double-density. Not so far, but maybe it just likes me.

2. -- NEWDOS/80 version 1.0 will NOT run unless some patches are applied. Yes, I realize that version 2.0 is the current version, but some of us still have a few things on 1.0 disks. About a week after installation, I needed a program on one of these disks. Having forgotten the experiences of another club

member, I tried to boot the disk. After the usual rapid back-and-forth stepping of the head, it went into "silent death" mode. I tried it a couple of more times just to be sure. Same results. Then I remembered that this was the same problem that caused the other club member to switch to an Aerocomp doubler. I'd rather fight that switch, so after a few hours of disassembly and tracing the DOS as it booted, I found the problem - it was getting switched into double-density mode! No wonder it got lost! A quick one byte patch, and it booted! Next a similar patch for the FORMAT routine was developed and now it works just like it used to (single density mode only, of course) as far as I can determine. So here's the patches:

```
SYS0/SYS,4,53     change 80 to 00.     (Trk 0/sec 9)
SYS0/SYS,11,70    change 80 to 00.     (trk 20/sec 1)
```

The 80H was being loaded into the sector register, which normally won't affect anything, but the R/S doubler uses that to switch to double density mode. A Percom doubler uses the command/status register to switch modes.

3. -- TRSDOS 2.8.00 finally arrives 6 months later! Back in December, in the "Notes, etc." column in 80-U.S., the problem with a single drive copy from single to double density was discussed. It was stated that version 2.8 would be available shortly, and new kits would come with it. BULL! I started checking with my local center, and calling Ft. Worth at least monthly. I wanted the most current DOS when I bought my kit. Nobody was aware of 2.8 locally, and Ft. Worth kept saying "another month". They didn't even acknowledge its existence until February. Finally, about a month ago, the local CSR made contact with somebody who knew about it. They said it had been available for a couple of months! When I got it, the date on the boot banner (looks like 1.3 with the computer picture) was DEC. 15, 1982! And 80-U.S. calls that "quick, excellent response". If that's quick, I'd hate to see slow!! By the way, you have to trade in your 2.7DD original to get the 2.8.00 version. End-of-editorial.

4. -- The following patches are by Tom Price via Compuserve, and are dated 1/24/82. There were five, but one was done in 2.8 so it is obsolete.

(1) These patches correct two errors in the PATCH command itself and this one MUST be applied first to allow proper operation of PATCH when patching 2.7DD system files.

```
PATCH *9 (R=6,B=235,F=07,C=00)
PATCH *9 (R=6,B=242,F=13,C=12)
```

(2) These patches correct a memory conflict between two of the system overlays during certain file I/O sequences. In addition, the 4430H call to the program loader was not preserving the IX register and this is also corrected.

```
PATCH *17 (R=1,B=4,F=321F45,C=C33451)
PATCH *17 (R=2,B=52,F=E5E5E5E5E5,C=321F45DDE5)
PATCH *17 (R=2,B=57,F=E5E5E5E5E5,C=CD0750DDE1)
PATCH *17
        (R=2,B=62,F=E5E5E5E5E5E5E5E5E5,C=F53E11321F45F1C9)
```

(3) These patches correct a problem with the $RAMDIR call which was causing the system to hang when retrieving single directory entries if the accessed directory slot was empty.

```
PATCH *14 (R=1,B=242,F=C0CB76,C=C3E54F)
PATCH *14
   (R=2,B=229,F=E5E5E5E5E5E5E5E5E5E5,C=C07EE6F0FE10C3F54E)
```

(5) This OPTIONAL patch removes the restriction in DEBUG which prevents accessing of any memory below 5400H.

```
PATCH *5 (R=1,B=207,F=54,C=00)
```

My research on these patches indicates that the problems Tom found do exist and that these patches fix them. Too bad Radio Shack didn't fix 'em but then we wouldn't have any fun finding the solutions ourselves.

5. -- As you saw in Tom's patches, the syntax of the PATCH operands is different for TRSDOS 2.7DD and 2.8. This is because all but one (the root - SYS0) of the 24 (That's right, 24!) overlays are in CIM format. There are no codes indicating where to load the code/data. It is simply read sector by sector into memory. This makes adding patches to the end possible with the PATCH command, because there is no "0202xxxx" transfer address loader code that prevents it. The overlays also load quicker, since the data is not buffered and processed byte-by-byte when loaded to the final location. Also, I developed variations of some existing patches from the Model III to do things my way. Here they are:

(a) Display error message text instead of number.
PATCH *4 (R=1,B=49,F=20,C=18)
PATCH *4 (R=1,B=63,F=20,C=18)

(b) Disable all password checking.
PATCH *2 (R=1,B=202,F=20,C=18)

[The following items are from the September/October 1983 issue of ELSE:]

1. -- Let's begin with a couple of loose bits. First, sometime in November or December I expect the sales of the standard Model 4 to slow considerably. Why? The answer is on page 2 of the new R/S RSC-10 catalog. It's the new Model 4P, with the "P" indicating portable! That's right, now even R/S has an Osborne/Kaypro clone. At $1799 for a 64K, 2 double-density 5 1/4" drives, 9 inch screen system, it will probably become the more popular version of the Mod 4. And with the 4 supporting that other DOS (CP/M) in addition to a real DOS (TRSDOS/LDOS 6.0), why would anybody want an Apple? Speaking of which, do you find the LISA commercial as odd as I do? I always thought that Altair was the first personal computer. And then the line about how everyone tried to make a better Apple. Most companies did a long time ago. My response to "have a computer or an Apple?" is "eat an apple while using a REAL computer". As you can tell, even with their faults, I'm a Tandy fan. Enuf said.

Second, when using a DOS without file date stamping (TRSDOS 2.3), I often use the "/EXT" to hold the month and day portions of the date. As you know, the first character of the extension must be alphabetic. By the way, 2.3 doesn't give any error if it isn't, it just ignores the bad extension! Therefore, by using a sequential letter to represent the month (A = Jan, B = Feb, C = Mar, etc.) followed by the day, the syntax requirements are met. A file created on 10/03/83 would use "J03" for example. Simple, but practical.

2. -- Now for the important stuff. As mentioned in my last column, I developed an extensive set of patches to TRSDOS 2.7/2.8 to allow a DIR of a non-system disk in drive 0. The 2.8 DOS allows you to copy from a 2.3 type disk to a 2.8 disk, but how do you find out what's on the 2.3 disk without rebooting in single density mode? The DIR command can detect and process a 2.3 type disk on drives 1, 2 or 3, but doesn't give you a chance to swap disks on drive 0. My patches allow you to specify a new parameter ("X") which displays disk mount messages as needed. If the DIR is aborted by use of the (BREAK) key or an error, a prompt for the system disk is issued. This code has been optimized as best as I can, and due to the size of the patch, I used the EDTASM program to make it easier. For those of you who are interested in assembler, here is the source code for the patches:

```
                00100 ; xxxxxx PATCH TO TRSDOS 2.7DD OVERLAY #21 (DIR) xxxxxx
                00110 ; x IT ALLOWS A DIR OF A NON-SYSTEM DISK IN DRIVE 0   x
                00120 ; x BY USING THE (X) PARAMETER, SIMILAR TO LDOS 5.1.3 x
                00130 ; x HOWEVER, DISK MOUNT MESSAGES DO NOT FLASH.        x
                00140 ; x THIS CODE IS FOR REFERENCE ONLY, SINCE THE PATCH  x
                00150 ; x COMMAND IS USED TO INSTALL THE CODE TO "#21".     x
                00160 ; x WRITTEN BY JOHN C MALLGREN  05/16/83 813-797-5125 x
                00170 ; x OPTIMIZED ON 07/24/83 -- COMPATIBLE WITH 2.8.00   x
                00180 ; xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
530E            00190         ORG     530EH   ; A 'LD HL,4200' INSTR IS USED
530E CD6058     00200         CALL    XPARM   ; AS A HOOK TO THE PARM (X) RTN
                00210
5853            00220         ORG     5853H   ; ADDR OF 7TH ENTRY IN TABLE
5853 58         00230         DEFB    'X'     ; CHANGE "A" TO "X"
                00240
5859            00250         ORG     5859H   ; ADDR OF PARM STORAGE AREA
5859 62         00260         DEFB    XFLAG&255 ; POINT TO OPERAND OF "LD DE,"
                00270
4467            00280 VDLINE  EQU     4467H   ; DISPLAY A LINE ON VIDEO
5860            00290         ORG     5860H   ; END OF TANDY CODE + 1 BYTE
5860 C5         00300 XPARM   PUSH    BC      ; SAVE DRIVE # IN C
5861 110000     00310         LD      DE,$-$  ; LOAD RESULT OF PARSER IN DE
5862            00320 XFLAG   EQU     $-2     ; USE OPERAND OF LD INSTR
5864 7A         00330         LD      A,D     ; TEST RESULT OF PARSER
5865 B3         00340         OR      E       ; 0000 = NO, FFFF = YES
```

```
5866 2825    00350        JR     Z,XEXIT ; 'X' PARM NOT SPECIFED, SO EXIT
5868 219858  00360        LD     HL,XNEX ; LOAD ADDR OF NORMAL EXIT
5868 22C354  00370        LD     (54C3H),HL ; INTO 'JP 5225' INSTR
586E 21A558  00380        LD     HL,XEEX    ; LOAD ADDR OF ERROR EXIT
5871 220852  00390        LD     (5208H),HL ; INTO 'JP 4409' INSTR
5874 21AD58  00400        LD     HL,XAEX    ; LOAD ADDR OF ABORT EXIT
5877 224852  00410        LD     (5248H),HL ; INTO THE THREE 'JP NZ,4030'
587A 22AA52  00420        LD     (52AAH),HL ; INSTR.  USED ONLY WHEN
587D 22B853  00430        LD     (53B8H),HL ; <BRK> IS ENTERED TO ABORT.
5880 CD9258  00440        CALL   XDSPIN     ; DSPLY "INSERT" ON VIDEO
5883 2ED4    00450        LD     L,XLSRCE&255 ; POINT TO "SOURCE"
5885 CD6744  00460        CALL   VDLINE     ; DISPLAY ON VIDEO
5888 2EC2    00470        LD     L,XLDISK&255 ; POINT TO "DISK..."
588A CD9F58  00480        CALL   XDSPMT     ; DSPLY MSG & WAIT FOR <CR>
588D 210042  00490 XEXIT  LD     HL,4200H ; DO OVERLAID INSTR
5890 C1      00500        POP    BC         ; RESTORE DRIVE 0
5891 C9      00510        RET               ; CONTINUE WITH TANDY CODE
             00520
5892 21B358  00530 XDSPIN LD     HL,XLINS ; POINT TO "INSERT" &
5895 C36744  00540        JP     VDLINE   ; EXIT THRU DSPLY MSG ON VIDEO
             00550 ;
5898 CD2552  00560 XNEX   CALL   5225H    ; DO OVERLAID INSTR FIRST
589B CD9258  00570 XN01   CALL   XDSPIN   ; DSPLY "INSERT",  THEN
589E 23      00580        INC    HL        ; POINT TO "SYSTEM DISK..."
589F CD6744  00590 XDSPMT CALL   VDLINE   ; DISPLAY ON VIDEO
58A2 C35C52  00600        JP     525CH    ; EXIT THRU WAIT FOR <CR>
             00610 ;
58A5 F5      00620 XEEX   PUSH   AF        ; SAVE ERROR NUMBER
58A6 CD9858  00630        CALL   XN01      ; DSPLY "INSRT SYS" & WAIT
58A9 F1      00640        POP    AF        ; RELOAD ERROR NUMBER
58AA C30944  00650        JP     4409H     ; EXIT THRU @ERRDSP
             00660 ;
58AD CD9858  00670 XAEX   CALL   XN01      ; DSPLY "INSRT SYS" & WAIT
58B0 C33040  00680        JP     4030H     ; EXIT TO @DOSABRT
             00690 ;
58B3 1D      00700 XLINS  DEFB   1DH       ; BACK TO LEFT SIDE
58B4 49      00710        DEFM   'Insert '
58BB 03      00720        DEFB   03H        ; <ETX>
58BC 53      00730 XLSYS  DEFM   'SYSTEM'
58C2 20      00740 XLDISK DEFM   ' Diskette <ENTER>'
58D3 0D      00750        DEFB   0DH
58D4 53      00760 XLSRCE DEFM   'SOURCE'
58DA 03      00770        DEFB   03H
             00780 ;
0000         00790        END
00000 TOTAL ERRORS
31113  TEXT AREA BYTES LEFT

VDLINE 4467 00280  00460 00540 00590     XLDISK 58C2 00740  00470
XAEX   58AD 00670  00400                 XLINS  58B3 00700  00530
XDSPIN 5892 00530  00440 00570           XLSRCE 58D4 00760  00450
XDSPMT 589F 00590  00480                 XLSYS  58BC 00730
XEEX   58A5 00620  00380                 XN01   5898 00570  00630 00670
XEXIT  588D 00490  00350                 XNEX   5898 00560  00360
XFLAG  5862 00320  00260                 XPARM  5860 00300  00200
```

Now that you know what the code looks like, let's translate it into patch format. Put these patches into a DO file so you can copy them to other disks. For increased readability, I've added a space between each byte in the Find and Change strings. DO NOT enter these when you key in the patch!!! For example, "21 00 42" MUST be entered as "210042" (without quotes of course).

In case you missed our last issue, there were two patches from Tom Price to fix a problem in the PATCH command. The problem still exists in 2.8.00 and my patches cannot be applied until it is corrected. So here they are again:

PATCH *9 (R=6,B=235,F=07,C=00)
PATCH *9 (R=6,B=242,F=13,C=12)

I would suggest that you put these two patches first in the DO file, and if they have been done already, they will be bypassed with a "String NOT found" error. Also, remember that if you make a mistake while BUILDing the DO file, it is relatively simple to correct by using the FILFIX utility on the completed file. However, you will have to enter the changes in hex, but that's no problem for a computer genius like you, right?? And if you enter some trailing blanks after the closing ")", you have room to add the character or two that were missed.

Let's take a closer look at how this DIR (X) modification works. In order to write it, I had to determine exactly how the current code worked and what parts of the existing code could be used so that the patches would take the least number of bytes. The overlay which processes the DIR command contains about 700 lines of code and is number 21 of 23 overlays. The best place I found to intercept processing was at 530EH, where HL is loaded with address of the I/O buffer for the reading of the boot sector. This instruction is overlaid in lines 190-200 with the CALL to my routine, and is then done in line 490 before returning to the original code. Since the prompt messages are not needed everytime a DIR is done, a yes/no type switch had to be incorporated.

One of the routines used in the DOS is a command parser, or scanner. This parser normally uses a table containing the keywords to be searched for, and a storage area to hold the coded argument value. In this case, there was an entry for the keyword "A", which was used in TRSDOS 2.3 to display the attributes for the files, but since that type of display is standard in 2.7, its only function was to prevent an error if it was accidentally used. We can use that entry for the new "X" keyword instead, so in lines 220-260 I establish the keyword value and the storage area instruction. This was done to avoid setting up a separate 2 byte storage area. As listed in the comments, lines 310-350 test the coded value to determine if the routine should be executed or bypassed. If "X" was specified, I then need to establish intercepts at the standard exit issued. This is done in lines 360-430.

The rest of the routine is fairly straightforward. What is different is that in lines 450 and 470, I only load the L register. The value in H remains constant, so why reload it? I use the "&" (AND) operator to trim the address to the low order byte. Also, note that I use the JP instruction to exit a CALLed subroutine. What actually happens is that the last instruction of the routine I jump to is a RETurn. This avoids doing another CALL/RET sequence. To further optimize the code, I broke the literals into pieces to avoid redundancy. The longer the literals, the better this works. Hope that this will be of some benefit to somebody.

PATCH *21 (A=2,B=014,F=21 00 42,C=CD 60 58)
PATCH *21
(A=7,B=083,F=41 20 20 20 20 20 5C, C=C5 11 00 00 7A B3 28 25)
PATCH *21
(A=7,B=096,F=E5 E5 E5 E5 E5 E5 E5 E5, C=C5 11 00 00 7A B3 28 25)
PATCH *21
(A=7,B=104,F=E5 E5 E5 E5 E5 E5 E5 E5, C=21 98 58 22 C3 54 21 A5)
PATCH *21
(A=7,B=112,F=E5 E5 E5 E5 E5 E5 E5 E5, C=58 22 0B 52 21 AD 58 22)
PATCH *21
(A=7,B=120,F=E5 E5 E5 E5 E5 E5 E5 E5, C=4B 52 22 AA 52 22 B0 53)
PATCH *21
(A=7,B=128,F=E5 E5 E5 E5 E5 E5 E5 E5, C=CD 92 58 2E D4 CD 67 44)
PATCH *21
(A=7,B=136,F=E5 E5 E5 E5 E5 E5 E5 E5, C=2E C2 CD 9F 58 21 00 42)
PATCH *21
(A=7,B=144,F=E5 E5 E5 E5 E5 E5 E5 E5, C=C1 C9 21 B3 58 C3 67 44)
PATCH *21
(A=7,B=152,F=E5 E5 E5 E5 E5 E5 E5 E5, C=CD 25 52 CD 92 58 23 CD)
PATCH *21
(A=7,B=160,F=E5 E5 E5 E5 E5 E5 E5 E5, C=67 44 C3 5C 52 F5 CD 9B)
PATCH *21
(A=7,B=168,F=E5 E5 E5 E5 E5 E5 E5 E5, C=58 F1 C3 09 44 CD 9B 58)
PATCH *21
(A=7,B=176,F=E5 E5 E5 E5 E5 E5 E5 E5, C=C3 30 40 1D 49 6E 73 65)
PATCH *21
(A=7,B=184,F=E5 E5 E5 E5 E5 E5 E5 E5, C=72 74 20 03 53 59 53 54)
PATCH *21
(A=7,B=192,F=E5 E5 E5 E5 E5 E5 E5 E5, C=45 4D 20 44 69 73 6B 65)
PATCH *21
(A=7,B=200,F=E5 E5 E5 E5 E5 E5 E5 E5, C=74 74 65 20 3C 45 4E 54)
PATCH *21
(A=7,B=208,F=E5 E5 E5 E5 E5 E5 E5 E5, C=45 52 3E 0D 53 4F 55 52)
PATCH *21 (A=7,B=216,F=E5 E5 E5,C=43 45 03)

(NOTE: Remember to remove all spaces when performing the patch; they are here only for improved readability.)

KBE PROGRAM ZAP by Tony Domigan - This will correct DVRIII64/CMD (from TAS KBE package) so that it will work on the model 4:

Location FF4EH the instruction	C30031 JP 3100H
Should be	C3FC33 JP 33FCH

A NEW OVERLAY MODULE FOR NEWDOS/80 VERSION 2,
JKL Routine with TRS-80 Graphic Blocks
for the EPSON RX80 / FX80 Printers

by Joachim Kelterbaum
(Frankenstr. 305, 4300 Essen 1, West Germany)

Many NEWDOS/80 users find it quite convenient to be able to get a screen dump to printer just by pressing the J-K-L keys. However, since the TRS-80 uses graphic blocks that are not usually included in the character generators of printers, it is not possible to make full use of this facility. Some printers, though, are able to do dot image graphics, and this provides a means to simulate the TRS-80 graphic blocks. In the following article, a way will be shown to implement this facility on NEWDOS/80 version 2 for the Epson RX80 / FX80 / MX80 III printers.

The first part of this article will deal with some general aspects of the function of the DOS overlay loader. In the second part I'll explain the JKL module itself. Part three will deal with a method of incorporating a self-written overlay into the system.

Function of the Overlay Loader

Certainly you know that NEWDOS/80 is a sophisticated operating system. Such degree of sophistication would by no means be possible if all of the system's functions were located in RAM at all times (you could do that, but there would be very little room left for user programs). The solution to this problem is the overlay technique. This is accomplished in the following manner:

There is only one part of the system (SYS0/SYS) permanently resident while NEWDOS/80 operates. In addition, there are approximately 20 overlay modules (SYS1/SYS to SYS21/SYS), of which only one is resident at a time. There are 2 overlay areas (the DOS overlay area at 4D00H-51FFH, and the secondary overlay area - used mostly by BASIC - at 5200H-6FFFH).

Each time a certain overlay is needed by the system, the DOS overlay loader - part of SYS0/SYS - will load this module to the overlay area (if that module is not loaded already). The module used before will simply be 'overlayed'. I suppose that you can well imagine that this particular function of the DOS is of vital importance to the system. This might be the reason why the function of the overlay loader is hardly documented in the manual. Obviously, the authors of NEWDOS/80 - as well as of the other systems - do not want us to fool around with the system.

On the other hand, this overlay area is an ideal place to put programs which do not use up any room in the user RAM and which are practically invisible to the system. But how can one make use of this area?

The answer to this is surprisingly simple. There are only 2 instructions needed: LD A,<code> ; RST 28H. <code> is a 1-byte constant which must meet the following conditions:

At least one of the high order three bits (bits 7, 6, or 5) must be set. The low order five bits (bits 4 through 0) tell the system where in the directory the needed module is located.

To make it easier to understand the following, you should turn on your computer and use the DFS (Display File Sectors) option of SUPERZAP to display File Relative Sector (FRS) 0 of DIR/SYS. Normally (depending on your particular PDRIVE setting) this is disk sector 170 on a single-sided, single-density 40 track disk.

FRS 0 is the GAT sector telling the system which tracks are formatted and which ones are already allocated to files. It also contains the disk name, date, and the AUTO command (if used). FRS 1 is the HIT sector. This is used by the system as a hash table for quickly finding a certain file on this disk. Starting from FRS 2 the directory entries begin. In the relative position 0 (top row) you always find BOOT/SYS. If you turn to FRS 3 you will find DIR/SYS in the top row having relative position 1. The top row of FRS 4 will show the file SYS0/SYS (relative position 2) and so on. Once you have reached FRS 9 of the DIR/SYS file (relative position 7 - in the case of a disk that was formatted using the standard PDRIVE parameter of DDGA=2, this is the end of the DIR/SYS file), you can start over with FRS 2. Now you count the entries in the second row, starting with relative position 8 (SYS6/SYS will be found there). Exactly these position numbers are the values you have to use in the constant <code> (low order 5 bits). As there will only be 5 bits decoded by the overlay loader you can only address up to 32 overlay modules this way (actually only 30, since BOOT/SYS and DIR/SYS occupy the first two positions). Let me give an example of an overlay call:

If you wanted to call SYS5/SYS (DEBUG), you'd find this entry at position 7 in the directory. If you remember to set one of the high order three bits (bit 7 in this case), your value for code should be 1000 0111B = 87H. So it is sufficient to execute the following instructions for an overlay call to DEBUG: LD A,87H ; RST 28H.

Exactly this method will be used to load our self-written JKL module. You will find the details in section three of this article.

I must say a few words on what conditions a /SYS module must meet in order to be loaded correctly by the overlay loader:

The module is a normal machine code routine which has to reside in one piece on disk (no additional extents allowed). The format of the module is the usual load file format which for example is produced by EDTASM. There has to be a start address stated in the END statement. This address will be jumped to after loading the file via RST 28H. The file need not be positioned within the overlay areas (of course, if you choose to have it load elsewhere, you must make sure that that area of memory is somehow "reserved", so that you overlay module won't overwrite another program already in memory). If you end your module with a RET instruction, a return will be made to the calling routine (i.e. the address on top of the stack).

If what you have just read encourages you to experiment, please, do so on a backup system!!! It is quite likely that you will get some errors in the beginning. These errors will sometimes be 'honoured' by a destroyed system. So, be careful!!!

Function of the JKL Module

You'll find the source code of the JKL module at the end of this article. First the printer is initialized, so that the standard settings of 10 Characters Per Inch and 12 dots linefeed will be used. After this the first line of the video RAM is loaded into the two buffers BUFTXT and BUFGRF. The buffer BUFTXT will now be modified to contain only printable ASCII codes - i.e. graphic codes above 7FH will be replaced by 20H and control codes 00H-1FH will be shifted up by 40H so they'll appear on the printer just like the characters being displayed by the TRS-80 character generator (the one normally supplied with the Radio Shack Model I lowercase modification, which duplicates the uppercase character set for ASCII codes 00H-1FH). In the buffer BUFGRF all codes below 80H will be replaced by 80H. This way we have a separation of text and graphics.

Now the program functions in the following manner: If a line only consisted of text codes, this text will be printed and a 12 dot linefeed will be done. If there were graphics, then first those graphics will be printed. As it is not possible to do a reverse linefeed with the MX80 III, only the upper 2 blocks of that row are printed. Then a 1 dot linefeed is done. After this the text is printed and a 7 dot linefeed is done. In a third phase the lower blocks of the graphics are printed and a 4 dot linefeed is done. This makes a full 12 dot high line of mixed graphics and text.

The whole procedure is repeated for all 16 video lines. Finally, the printer is reset to normal again and a return to the calling program (usually DOS or BASIC) is done.

Though this procedure of a JKL-dump sounds quite complicated, it works surprisingly fast.

Hints for Installation of the Module to the NEWDOS/80 System

After you have typed in the source using EDTASM or a similar program, save it as SYSJKL/CMD on a working diskette. Now, make a copy of your (naked) system on a different diskette (COPY,0,1,,FMT,CBF,/SYS or similar). Boot this system. Now you have to install a file at a particular position in the directory (see section 1 of this article). We will use position number 1DH = 29 in this example. To create a file entry in position 1DH proceed as follows:

(The method described here is not the most elegant one, but it's the safest)

Create several files using the CREATE command (CREATE S0:0 ; CREATE S1:0 etc.). Now use SUPERZAP's DFS option again to display DIR/SYS. Find the entry at position 29 (this will be found starting at FRS 7, byte 60H). Write down the name of this entry. If there is no entry at this position, continue creating files.

Now, boot your system again and rename that file at position 29 to SYSJKL/SYS.

Finally, you can purge all those files that are not needed any longer. If you execute a DIR 0 now, you'll find that SYSJKL/SYS is present as a normal visible file. This does not look very professional for a /SYS module. Use SUPERZAP again to change the first 2 bytes of the directory entry of SYSJKL/SYS to 5FH, 20H. Now, your file will only be displayed by DIR 0 /SYS. Now, copy the file SYSJKL/CMD to SYSJKL/SYS onto your new system diskette and you're all set.

The /SYS module is installed now, but your system will not recognize that it is there. In the Model I version of NEWDOS/80, the normal JKL routine is located in SYS3/SYS, FRS 4, starting at byte 96H (byte 73H on the Model III version of NEWDOS/80). At this place we apply the call to our own module: LD A,9DH ; RST 28H, which is 3EH, 9DH, EFH. Once you have zapped those three bytes, your module will work.

Boot your system and try it. Don't forget to copy BASIC/CMD as well as all other programs you need to your new system.

Happy JKL-ing!

Joachim Kelterbaum

EDITOR'S NOTE: For further information on creating your own /SYS files under NEWDOS/80, I refer you to the article on that subject in the previous issue of Northern Bytes, also the additional comments in "The Exterminator" column elsewhere in this issue. Also, Joachim has provided me with two versions of this program - the one below, for use with Epson and compatible printers, and one for use with Centronics 739 printers. Since the 739 version is not commented, I am not reprinting it here, but you may obtain a copy of the source code listing by sending a self-addressed stamped envelope to me (use the Sault Ste. Marie address for this, not the Lansing address). Here's the source code listing for the Epson-compatible version of the program:

```
00100 ;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
00110 ;x          JKLSYS/SRC                 x
00120 ;x  a NEWDOS 80/2 System-Modification  x
00130 ;x      of JKL with Graphic-Blocks     x
00140 ;x         for EPSON-Printers          x
00150 ;x of Types MX80 II, III , RX80, FX80  x
00160 ;x                                     x
00170 ;x        Joachim Kelterbaum           x
00180 ;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
00190         ORG     4D00H   ;start system overlay area
00200 START   EXX             ;save registers
00210         CALL    INIT    ;initialize printer
00220         LD      BC,200H ;delay loop
00230         CALL    60H     ;
00240         LD      A,0DH
00250         CALL    PRT
00260         LD      A,0DH   ;2 LF's
00270         CALL    PRT     ;
00280         LD      HL,3C00H-64     ;linepointer
                              video-RAM
00290         LD      (ZEIAD),HL
00300         LD      BC,16   ;16 lines
00310 ZEILE   PUSH    BC      ;start of line loop
00320         LD      BC,64
00330         LD      HL,(ZEIAD)
00340         ADD     HL,BC
00350         LD      (ZEIAD),HL      ;update linepointer
00360         LD      DE,BUFTXT
00370         LDIR            ;transfer line to BUFTXT
00380         LD      BC,64
00390         LD      DE,BUFGRF
00400         LD      HL,(ZEIAD)
00410         LDIR            ;transfer line to BUFGRF
00420         LD      HL,BUFGRF
00430         LD      B,64
00440 BERGRF  LD      A,(HL)  ;load char from BUFGRF
00450         CP      80H
00460         JR      NC,SKIP
00470 N360    LD      A,80H   ;if no graphics,so
00480         LD      (HL),A  ;replace by 80H
00490 SKIP    INC     HL      ;OK - go on
00500         DJNZ    BERGRF
00510         LD      HL,BUFTXT
00520         LD      B,64
00530         LD      A,0
00540         LD      (FLG),A ;FLG=0,if no graph. in line
00550 BERTXT  LD      A,(HL)  ;load char from BUFTXT
00560         CP      20H
00570         JR      NC,NOCTL;intercept CNTL codes
00580         ADD     A,40H   ;replace by correspond. ASCII
00590         JR      CTLSP
00600 GRFK    LD      A,20H
00610 CTLSP   LD      (HL),A
00620 OK      INC     HL
00630         DJNZ    BERTXT
00640         JR      WEITER
00650 NOCTL   CP      80H
00660         JR      C,OK
00670         LD      A,1     ;if graphics,so set FLG
00680         LD      (FLG),A
00690         JR      GRFK    ;replace by 20H in BUFTXT
00700 WEITER  CALL    TEST    ;if no graphics,so print
00710         JP      Z,FERTIG;and --> FERTIG
00720         CALL    ESCS    ;initialize graphics-mode
00730         LD      B,60
00740         LD      A,0
00750 INDENT  CALL    PRT
00760         DJNZ    INDENT  ;print 60 bytes of 80H
                              (indent)
00770         LD      B,64
00780         LD      HL,BUFGRF
00790 Z1GR    LD      E,0     ;clear E
00800         LD      A,(HL)  ;load char from BUFGRF
00810         LD      D,A     ;save to D
00820         BIT     0,A     ;bit 0 set ?
00830         JR      Z,NO1   ;no, go --> NO1
00840         LD      A,E     ;yes get E
00850         OR      0F0H    ;set top 4 bits
00860         LD      E,A     ;save E
00870         LD      A,D     ;get char
00880 NO1     BIT     2,A     ;bit 2 set ?
00890         JR      Z,NO2   ;no, go --> NO2
00900         LD      A,E     ;yes get E
00910         OR      0FH     ;set lower 4 bits
00920         LD      E,A     ;save E
00930 NO2     LD      A,E     ;get E
00940         CALL    PR3     ;print 3 mal (1/2
                              characterwidth)
00950         LD      E,0     ;...
00960         LD      A,D
00970         BIT     1,A
00980         JR      Z,NO13
00990         LD      A,E
01000         OR      0F0H
01010         LD      E,A
01020         LD      A,D
01030 NO13    BIT     3,A
01040         JR      Z,NO33
01050         LD      A,E
01060         OR      0FH
01070         LD      E,A
01080 NO33    LD      A,E
01090         CALL    PR3     ;like above, but with bits 1
                              and 3
01100         INC     HL
01110         DJNZ    Z1GR
01120         LD      A,27
01130         CALL    PRT
01140         LD      A,'A'
01150         CALL    PRT
01160         LD      A,1
01170         CALL    PRT     ;1 dot linefeed
01180         LD      A,0DH
01190         CALL    PRT     ;LF
01200         LD      B,10
01210         LD      A,20H
01220 IND2    CALL    PRT
01230         DJNZ    IND2    ;10 blanks indent
01240         LD      HL,BUFTXT
01250         LD      B,64
01260 ZTXOUT  LD      A,(HL)
01270         CALL    PRT
01280         INC     HL
01290         DJNZ    ZTXOUT  ;print textline
01300         LD      A,27
01310         CALL    PRT
01320         LD      A,'A'
01330         CALL    PRT
01340         LD      A,7
01350         CALL    PRT     ;7 dots linefeed
01360         LD      A,0DH
01370         CALL    PRT     ;LF
01380         CALL    ESCS    ;initialize graphics-mode
01390         LD      B,60    ;leave 2 graphics-blocks
01400         LD      A,0     ;just like above
01410 IND6    CALL    PRT
01420         DJNZ    IND6
01430         LD      B,64
01440         LD      HL,BUFGRF
01450 Z2GR    LD      A,0
01460         LD      E,A
01470         LD      A,(HL)
01480         LD      D,A
01490         BIT     4,A
01500         JR      Z,ND11
01510         LD      A,E
01520         OR      0F0H
01530         LD      E,A
01540 ND11    LD      A,E
01550         CALL    PR3
01560         LD      E,0
01570         LD      A,D
01580         BIT     5,A
01590         JR      Z,ND15
01600         LD      A,E
01610         OR      0FH
01620         LD      E,A
01630 ND15    LD      A,E
01640         CALL    PR3
01650         INC     HL
01660         DJNZ    Z2GR
01670         LD      A,27
01680         CALL    PRT
01690         LD      A,'A'
01700         CALL    PRT
01710         LD      A,4
01720         CALL    PRT     ;4 dots linefeed
01730         LD      A,0DH
01740         CALL    PRT     ;LF
01750 FERTIG  POP     BC
01760         DEC     BC
01770         LD      A,B
01780         OR      C
01790         JP      NZ,ZEILE        ;end of line loop
01800         CALL    INIT    ;initialize printer
01810         EXX             ;restore registers
01820         XOR     A       ;set Z flag (no error cond)
01830         RET             ;back to system
01840 ;
01850 TEST    LD      A,(FLG) ;UP
01860         CP      0       ;if only text in line ,so
                              print
01870         RET     NZ      ;else return
01880         LD      B,10
01890         LD      A,20H
01900 LEER    CALL    PRT
```

```
01910          DJNZ    LEER
01920          LD      HL,BUFTXT
01930          LD      B,64
01940 ZX       LD      A,(HL)
01950          CALL    PRT
01960          INC     HL
01970          DJNZ    ZX
01980          LD      A,27
01990          CALL    PRT
02000          LD      A,'2'
02010          CALL    PRT     ;12 dots linefeed
02020          LD      A,00H
02030          CALL    PRT     ;LF
02040          XOR     A       ;set Z flag
02050          RET
02060 ;
02070 INIT     LD      A,27    ;UP- initialize printer
02080          CALL    PRT
02090          LD      A,'R'
02100          CALL    PRT
02110          LD      A,0
02120          CALL    PRT
02130          LD      A,27
02140          CALL    PRT
02150          LD      A,'@'
02160          CALL    PRT
02170 ;        LD      A,27    ;country selection GERMANY
02180 ;        CALL    PRT     ;leave out in American
                               version
02190 ;        LD      A,'R'
02200 ;        CALL    PRT
02210 ;        LD      A,2
02220 ;        CALL    PRT
02230          RET
02240 ;
02250 ESCS     LD      A,27    ;UP- initialize
                               graphics-mode
02260          CALL    PRT
02270          LD      A,'K'
02280          CALL    PRT
02290          LD      A,188
02300          CALL    PRT
02310          LD      A,1
02320          CALL    PRT
02330          RET
02340 ;
02350 PR3      PUSH    BC      ;UP- print char 3 times
02360          LD      B,3
02370 PTG      CALL    PRT
02380          DJNZ    PTG
02390          POP     BC
02400          RET
02410 ;
02420 PRT      PUSH    HL      ;print char
02430          LD      HL,37E8H
02440 CHK      BIT     7,(HL)
02450          JR      NZ,CHK
02460          LD      (HL),A
02470          POP     HL
02480          RET
02490 ;
02500 FLG      DEFB    0
02510 ZEIAD    DEFW    0
02520 BUFTXT   DEFS    64
02530 BUFGRF   DEFS    64
02540 ;
02550          END     START
```

NEWDOS/80'S PARAMETER SCANNER - Copyright by Greg Small - 23 February 1984!

Newdos/80 Version 2.0 has a parameter scanner that can be of considerable use to assembly language programmers.

I have never seen this documented and in my disassemblies of Newdos/80 have found a considerable number of CALLs to this routine. I decided that this information would be helpful to a number of Newdos/80 users and decided to make it available to Jack and Charley. This may also stimulate some interest in the proposed Newdos/80 users group as outlined in Volume 5, Number 3 of Northern Bytes.

This routine is used to parse commands with optional parameters such as "SYSTEM,0,AY=N,AZ=N" or "VERIFY,Y". This later command could also be entered as "VERIFY Y" or as "VERIFY Y". The command could also be incorrectly entered as "VERIFY,,Y" or "VERIFY, Y" or "VERIFY,T".

The Newdos/80 command interpreter's positioning of the HL register pair is explained in ZAP 063 Part 3 (Model I) and ZAP 057 Part 3 (Model III).

The routine in question is located at 4CD5H for the Mod I and at 4C7AH for the Mod III. It is 24 bytes long and can be called with the following setup!

```
LD      HL,TEXT     ;Point HL to command string
                    ;Unnecessary if HL has not been changed
                    ;since the program loaded as HL will point
                    ;to TEXT at entry
CALL    ROUTINE     ;Use routine in question
```

Following is a summary of the Entry and Exit conditions!

```
Entry! HL => ","
Exit!  HL => Byte after ","
       A  =  34H
       Z  =  reset
       C  =  set

Entry! HL => Space or series of spaces
Exit!  HL => first non-space
       A  =  34H
       Z  =  reset
       C  =  reset

Entry! HL => Carriage return
Exit!  HL => Carriage return
       A  =  0DH (Carriage return)
       Z  =  set
       C  =  reset

Entry! HL => Byte other than one of the above
Exit!  HL => No change
       A  =  34H
       Z  =  reset
       C  =  set
```

The following commented disassembly of this routine is for the Mod I. The Mod III version is identical but starts at 4C7AH.

```
4CD5 7E      LD      A,(HL)      ;Get byte pointed to by HL into A
4CD6 FE0D    CP      0DH         ;Is it a carriage return
4CD8 C8      RET     Z           ;YES - return with Z set
4CD9 7E      LD      A,(HL)      ;NO - Get byte again
4CDA FE2C    CP      2CH         ;Is it a comma
4CDC 23      INC     HL          ;Point HL to next byte
4CDD 280A    JR      Z,4CE9H     ;YES - JR to prepare to bail out
4CDF FE20    CP      20H         ;NO - Then is it a space
4CE1 2B      DEC     HL          ;Point HL back to original byte
4CE2 37      SCF                 ;Set Carry flag
4CE3 2005    JR      NZ,4CEAH    ;NO - JR to prepare to bail out
4CE5 23      INC     HL          ;YES - point to next byte again
4CE6 BE      CP      (HL)        ;Is this byte same as A (space)
4CE7 28FC    JR      Z,4CE5H     ;YES - JR to test for space again
4CE9 B7      OR      A           ;Reset CARRY flag
4CEA 3E34    LD      A,34H       ;Set A to 34H
4CEC C9      RET                 ;Return to caller
```

I want to continue with this type of disassembly and would like to hear from any of you who have disassembled and commented similar routines. You may contact me at the following address!

Greg Small
Box 607
Stouffville, Ontario, CANADA
L0H 1L0

or you can leave a message on my bulletin board. New users are limited to 15 minutes due to the high caller traffic we get from Toronto so you should have your message ready for uploading. The board is a TBBS type so you may already be familiar with the message uploading formats. The number for the board is (416) 640-3434 and the parameters are 8N1 - 300 baud only. Please select <R>egister as your first choice so I can set you up for special access for Northern Bytes readers.

[For those of you that missed last month's issue of NORTHERN BYTES, Greg is attempting to start an international NEWDOS/80 user's group. If you're a NEWDOS/80 user, be sure and drop Greg a line or dial up his BBS, and let him know of your interest].

FORTH BULLETIN BOARD SYSTEMS by Paul Snively - For those who are interested, there are two BBS's that I have found that offer support for Forth fanatics.

The first is at (206) 759-0615. The computer is a TRS-80 Model I running QFORTH 1.8A. This BBS is written in Forth, and it has a tree structure for its messages. It takes a little getting used to, so for the first attempt the caller might want to download the help files. Incidentally, this board supports 1200 baud communication (a lifesaver!).

The other is at (206) 756-0448. This computer is also a TRS-80, but the software is TBBS. However, this is the only TBBS that I know of that has an active Forth SIG. It also has limited download capability in Forth. This system also supports 1200 baud.

A PATCH OF A PATCH by Nate Salsbury (610 Madam Moore's Lane, New Bern, North Carolina 25860)!

In the October, 1981 edition of '80 Microcomputing', an article by Arne Rhode (Pilevej 31, 7600 Struer, Denmark) outlined a program to transfer EDTASM-Plus (by Microsoft) from its Model I cassette version to a Model I disk environment. You could then load in your cassette source programs and store them to and retrieve them from disk. Also, new source could be saved/retrieved from disk. I had (then) recently moved from a cassette Model I to a 2-disk Model III and was VERY anxious to salvage my miles of tapes and months of typing source text. I blithely followed Arne's excellent instructions on how to proceed.

After successfully getting my Model I (cassette) EA+ on to my Model III disk, I happily started to 'salvage' my first Model I tape. All went smoothly until the LAST BYTE and then the keyboard locked up and the screen jittered around with some miscellaneous garbage on it! After some excellent hints from Jack Decker and Bruce Hansen, I finally traced the source of my problem and evolved this Patch of a Patch (lines 2000 to 2560 of this listing). The difficulty came about because of the different cassette routines in the Model III ROM - in particular 235H (read a byte) and 1F8H (turn off cassette). A brief review of my findings and 'fix' follow.

After EA+ finishes its initialization, it jumps to a routine starting at 44CCH. The first four lines here read!

```
44CCH   LD      SP,4380H
44CFH   CALL    01F8H           ;Turn off cassette
44D2H   LD      BC,44CCH
44D5H   PUSH    BC
```

This reveals two points! (1) The 'final' RET address on the stack is 44CCH and (2) the first action after getting there is to 'turn off the cassette' - in other words, this is always the first thing EA+ does when it recycles to 44CCH to await your next command. I was finally led to tracing out the 01F8H routine in ROM.

At 01F8H there is a jump to 300CH and from there a jump to 31C0H where the routine really starts. The first two instructions seem to create most of the problem!

```
31C0H   LD      A,(4213H)       ;Normally with TRSDOS => 04H
31C3H   OUT     (0E0),A
```

followed by some routine housekeeping and, finally, at 31CFH!

```
31CFH   EI                      ;Enable interrupts
31D0H   RET
```

To this day, I'm not sure WHY that OUT (0E0H),A at 31C3H caused my problem (along with the EI at 31CFH). However, those two WERE the cause so I fixed them. I also found that EA+ also used the byte at 4213H (which distressed BASIC) so I had to work out a "time sharing" arrangement.

I will not describe any aspect of Arne Rhode's original patch since this was covered thoroughly in the referenced "80 Microcomputing" article. The reader is advised to review that text. It will cover this listing through line 1840 (I have made several modest changes to fit my EA+ version 1.07. More on that later. Also, my line numbers differ from the article but the instructions are the same).

My patch accomplishes two main purposes! (1) Save the byte at 4213H under EA+ and give it back to BASIC at the appropriate times and (2) DISable interrupts after every call to the 1F8H routine. Along the way, I added a third item which was to return to the EA+ '*' prompt after using BREAK to stop a cassette load (rather than return to BASIC). Please note that my references to EA+ addresses refer to my version, 1.07. If you have a different version, the addresses you use may vary slightly but they SHOULD be close to those I cite here (e.g. note the difference in my ORG 6486H and Arne's article which ORGs at 646CH).

When EDTASM starts to Load a source tape, we get the READY CASSETTE prompt. In version 1.07 this is accomplished by a CALL 4408H instruction. To find your equivalent, disassemble the Command Jump table (in version 1.07 this lies between 4659H and 469AH). Find the ASCII 'L' and then disassemble EA+ at the address pointed at by the two bytes immediately following the 'L'. In Version 1.07 that was 4D0EH. Here I found the following code!

```
4D0EH   CALL    4FB1H           ;Of no concern here
4D11H   LD      E,0D3H          ;Identifying byte at the start
                                ;  of an EDTASM source file
4D13H   CALL    4FFEH           ;Finds the start of your source
                                ;  file
```

At 4FFEH I found!

```
4FFEH   CALL    4408H
```

Disassembly of the 4408H routine showed that THIS was the routine which puts READY CASSETTE on the screen. Since this is just before we fire up the cassette, I used TASMON to search EA+ and Z-BUG for similar calls and discovered them at 4FFEH (as noted) and also at 4FD4H. At these two spots I replaced 4408H with the address of my CASSON label (see line 2000). There, I call the routine to print READY CASSETTE, set the cassette speed to 'L', change the BREAK vector to EDTASM's warm start, and replace the byte in register E (0D3H - the first byte of a source tape which was in E when EA+ gets to CASSON). Then, it's back to EA+ for continuation of source tape loading.

Since I wanted to fiddle with the "cassette off" routine at 1F8H, I also searched EA+ and Z-BUG for CALL 1F8H and found it (in version 1.07) at 44CFH, 4D1FH, 650DH and 71C6H (the last two are in Z-BUG). I replaced the CALL 1F8H at each location with a CALL to my CASOFF address (line 2260). Here, I save the byte at 4213H which 'belongs' to EA+, and replace it with the 04H byte which is normally found at that address by the ROM routine. Then, after a CALL 1F8H to turn off the cassette, I immediately DISable the interrupts again, restore the usual BREAK vector and get back to EA+ for whatever.

This patch allowed me to salvage all those Model I source tapes and get them on to my disk. Let me hasten to warn any reader that I found that SOMETIMES, the 'Find a String' routine of EA+ would NOT work properly. I cannot guarantee that everything else is OK because shortly after doing this, I became interested in other work and, eventually, got the Model III disk version of EA+ and discontinued the project. However, for my purposes, the major task of salvage had been accomplished. I was subsequently able to transfer these files over to my 'new' EA+ and that's what I started out to do.

Several comments are appropriate here regarding this patch. Of necessity I had to add it at the end of Arne Rhode's program so that source texts would not overwrite the patch. This, in turn, required a modification to his EQU for BFSTAD (at line 140 of his listing in the original article. It is line 470 in this listing). The value you want there is the location shown in line 2520 of this listing PLUS 0E80H - the amount of the offset from 4380H to 5200H (Note! Arne suggests saving everything from 4000H up - if you follow that path you will have to add 1200H instead of 0E80H). Since different versions of EA+ apparently differ in length, you'll have to experiment to determine where to locate your version of my patch.

A final note on Arne Rhode's article - I found it necessary to insert line 1630 XOR A (to clear the carry flag). Because of differences in my version of EA+, I started the routine shown in his lines 1200-1260 at 7330H. My lines 1790 - 1820 calculate the length of the program to be moved and then relocates it to its normal position. Since EA+ clobbers DOS, I saw no point in saving RAM from 4000H to 437FH. This has two benefits! (1) The transferred source text (when you save to disk) can be located lower in RAM, thus allowing longer assemblies and (2) Less disk space is used. With this background, my TRANSFER address of the LOAD module of my source text is 7330H + (5200H - 4380H) = 81B0H (see lines 1390 - 1420).

I also had to change EA+ at 4389H and 438FH where it CALLs 0215H. I changed these to 'CALL 0212H' and that was the final touch. Since my interest was to salvage my old cassette source tapes (but NOT to create new ones) I never explored my revised setup's ability to write source text to tape (since it could now store it on disk). Also, I did not do Stand Alone Z-BUG because TASMON does all the same things and much more. However, the procedure would be similar once you locate Z-BUG's command jump table. In my version of Stand Alone Z-BUG , this is located from 4FEBH to 501AH with the "L" command located at 500FH.

One last note! EA+ replaces the RST 20, 28 and 30 instructions with its own routines. Don't try to use Z-BUG on programs that use any of these three RST instructions - either directly or via ROM routines - not unless you enjoy reading your "XYZ DOS READY" message lots of times!

I wish to thank Jack Decker and Bruce Hansen for their extensive time with me on the telephone as I bumbled about trying to unravel this mystery. This article could not have been written without their comments and suggestions which finally pointed me to the solution. My other silent co-contributor was my monitor, TASMON. It even allowed me to single-step the ROM routines!

```
                00100 ;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                00110 ;x                A PATCH OF A PATCH                      x
                00120 ;x                                                        x
                00130 ;xTRANSFERRING MICROSOFT'S EDTASM-PLUS MODEL I CASSETTE    x
                00140 ;xPROGRAM TO A MODEL III DISK (WITH THE ADDED VIRTUE OF    x
                00150 ;xSAVING SOURCE TEXT TO DISK).                            x
                00160 ;x                                                        x
                00170 ;x                NATE SALSBURY                           x
                00180 ;x                610 MADAM MOORE'S LANE                  x
                00190 ;x                NEW BERN, NC   28560                    x
                00200 ;x                                                        x
                00210 ;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                00220 ;x                                                        x
                00230 ;xTHE FIRST PART OF THIS 2-PATCH LISTING (LINES 470 TO     x
                00240 ;x1840) APPEARED IN THE OCTOBER '81 EDITION OF '80 MICRO   x
                00250 ;xCOMPUTING', PAGE 344. THE AUTHOR WAS ARNE RHODE. THE     x
                00260 ;xREADER IS REFERRED TO THE ARTICLE FOR A DETAILED DIS-    x
                00270 ;xCUSSION OF THAT PART OF THIS LISTING. IT HAS BEEN RE-    x
                00280 ;xPEATED HERE FOR CONVENIENCE IN REVIEWING MY ADDITIONS.   x
                00290 ;x                                                        x
                00300 ;xTHE ORIGINAL ARTICLE DISCUSSES VARIOUS TECHNIQUES FOR    x
                00310 ;xINITIALLY GETTING THE CASSETTE VERSION TO YOUR DISK.     x
                00320 ;x                                                        x
                00330 ;xSEVERAL CHANGES WERE MADE BY ME TO ACCOMODATE MY MACH-   x
                00340 ;xINE AND MY VERSION OF EDTASM-PLUS. THESE CHANGES ARE     x
                00350 ;xDISCUSSED IN THE TEXT OF THIS ARTICLE.                  x
                00360 ;x                                                        x
                00370 ;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                00380 ;
                00390 ;       ++++++++++++++++++++++++++++++
                00400 ;       +IMPORTANT  NOTE+
                00410 ;       ++++++++++++++++++++++++++++++
                00420 ;
                00430 ;TO USE THESE PATCHES YOU GIVE UP THE EDTASM 'QUASH'
                00440 ;FEATURE. IT IS REPLACED BY COMMANDS TO EXIT EDTASM
                00450 ;AND STORE YOUR SOURCE TEXT ON DISK.
                00460 ;
8210            00470 BFSTAD  EQU     8210H           ;START OF 'DUMP' BUFFER
                00480                                 ; WHICH IS 7390H (BUFST)
                00490                                 ; + 0EB0H OFFSET FROM
                00500                                 ; 4380H TO 5200
8212            00510 BFENAD  EQU     BFSTAD+2        ;END OF ACTUAL TEXT=8212H
8214            00520 BUFFER  EQU     BFENAD+2        ;TEMP STORGE BUFFER=8214H
                00530 ;
                00540 ;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                00550 ;       CHANGE THIS EQU FOR YOUR MACHINE IF NEEDED
                00560 ;
FFFF            00570 MEMEND  EQU     0FFFFH
                00580 ;
                00590 ;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                00600 ;
                00610 ;YOU MAY HAVE TO 'ORG' AT DIFFERENT ADDRESS. THE ARTICLE
                00620 ;IN '80 MICROCOMPUTING' HAS INFORMATION ON HOW TO FIND
                00630 ;THE RIGHT PLACE IN YOUR EDTASM-PLUS.
                00640 ;
6486            00650         ORG     6486H           ;START OF 'QUASH' SEQ
                00660 ;
                00670 ;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
6486 2A3242     00680         LD      HL,(4232H)      ;END OF EDTASM TEXT PTR
6489 E5         00690         PUSH    HL              ;SAVE END ADDRESS
648A ED5B3042   00700         LD      DE,(4230H)      ;START OF EDTASM TEXT PTR
648E D5         00710         PUSH    DE              ;SAVE START ADDR
648F 23         00720         INC     HL              ;MOVE PAST TWO 0FFH
6490 23         00730         INC     HL                ;BYTES AT END OF TEXT
6491 E5         00740         PUSH    HL              ;SAVE THIS POINT TOO
6492 23         00750         INC     HL              ;JUST PAST END OF ALL TXT
6493 AF         00760         XOR     A               ;CLEAR CARRY
6494 ED52       00770         SBC     HL,DE           ;FIND LENGTH TO MOVE
6496 44         00780         LD      B,H
6497 4D         00790         LD      C,L             ;BYTE COUNT TO BC
6498 211382     00800         LD      HL,BUFFER-1     ;DEST ADDR -1..
649B 09         00810         ADD     HL,BC             ; + LEN = END ADDRESS
649C EB         00820         EX      DE,HL           ;DE HAS BUFFER END ADDR
649D E1         00830         POP     HL              ;END OF EDTASM SOURCE
649E D5         00840         PUSH    DE              ;SAVE END OF BUFFER ADDR
649F EDB8       00850         LDDR                    ;MOVE TEXT TO BUFFER
64A1 D1         00860         POP     DE              ;END OF BUFFER AGAIN
64A2 E1         00870         POP     HL              ;START OF EDTASM SOURCE
64A3 221082     00880         LD      (BFSTAD),HL     ;SAVE EDTASM START ADDR
64A6 E1         00890         POP     HL              ;END OF EDTASM SOURCE
64A7 221282     00900         LD      (BFENAD),HL     ;SAVE EDTASM END ADDR
64AA 21CE72     00910         LD      HL,DMPADR       ;SCREEN TXT FOR 'DUMP'
64AD 7A         00920         LD      A,D             ;MSB END OF BUFFER ADDR
64AE CDB472     00930         CALL    CNVL            ;TOP 4 BITS TO HEX
64B1 7A         00940         LD      A,D
64B2 CDB872     00950         CALL    CNVR            ;LOW 4 BITS TO HEX
64B5 7B         00960         LD      A,E             ;LSB OF END OF TEXT
64B6 CDB472     00970         CALL    CNVL            ;TOP 4 BITS TO HEX
64B9 7B         00980         LD      A,E
64BA CDB872     00990         CALL    CNVR            ;LOW 4 BITS TO HEX
64BD 219572     01000         LD      HL,TEXT
64C0 CD3245     01010         CALL    4532H   ;TEXT TO SCREEN VIA EDTASM
64C3            01020 WTSPC   EQU     $
64C3 3A4038     01030         LD      A,(3840H)       ;KEYBOARD MEMORY
64C6 E680       01040         AND     80H             ;ISOLATE 'SPACE'
64C8 28F9       01050         JR      Z,WTSPC         ;ONLY GO ON 'SPACE'
64CA C30000     01060         JP      0               ;RE-BOOT WHEN DONE
                01070 ;
                01080 ;  NOW, GO TO END OF ACTUAL EDTASM PROGRAM AND ADD MORE
                01090 ;
7284            01100         ORG     7284H           ;JUST PAST END OF EDTASM+
                01110 ;
                01120 ;CONVERT NIBBLE TO A HEX CHARACTER
                01130 ;
7284            01140 CNVL    EQU     $
7284 0F         01150         RRCA
7285 0F         01160         RRCA
7286 0F         01170         RRCA
7287 0F         01180         RRCA                    ;MOVE MSB TO LSB
7288            01190 CNVR    EQU     $
7288 E60F       01200         AND     0FH             ;REMOVE UPPER HLF
728A F630       01210         OR      30H             ;TEST FOR NUMERIC
728C FE3A       01220         CP      3AH             ;CHECK FOR ALPHA
728E 3802       01230         JR      C,HEXOK         ;IT'S A NUMBER
7290 C607       01240         ADD     A,07H           ;CONVERT TO A-F
7292            01250 HEXOK   EQU     $
7292 77         01260         LD      (HL),A          ;STORE IN SCREEN MSG
7293 23         01270         INC     HL              ;TO NEXT SPOT IN MSG
7294 C9         01280         RET
7295 0A0A       01290 TEXT    DEFW    0A0AH           ;TWO LINE FEEDS
7297 52         01300         DEFM    'RECORD FOLLOWING INFORMATION'
    45 43 4F 52 44 20 46 4F 4C 4C 4F 57 49 4E 47 20
    49 4E 46 4F 52 4D 41 54 49 4F 4E
72B3 0A0A       01310         DEFW    0A0AH           ;TWO LINE FEEDS
72B5 44         01320         DEFM    'DUMP FILESPEC NAME '
    55 4D 50 20 46 49 4C 45 53 50 45 43 20 4E 41 4D
    45 20
72C8 38         01330         DEFM    '8210H'         ;START OF RELOCATED SRC
    32 31 30 48
72CD 20         01340         DEFB    20H
72CE 58         01350 DMPADR  DEFM    'XXXXH'         ;PROGRAM FILLS THIS IN
    58 58 58 48
                01360                                   ;AS 'END' OF DUMP
72D3 20         01370         DEFB    20H
                01380 ;
                01390 ;TRANSFER ADDRESS FOLLOWS. CALCULATED=> PROG SHIFTED BY
                01400 ;5200H-4380H=E80H. THE 'TRANSFER' SECTION IS NOW 'ORG' AT
                01410 ;7330H, THEREFORE, 7330H+E80H=8180H - ADDRESS FOR TRANS-
                01420 ;FER ROUTINE WHEN PROGRAM IS FIRST LOADED IN FROM DISK.
                01430 ;
72D4 38         01440         DEFM    '8180H'         ;ENTRY POINT TO MOVE ALL
    31 42 30 48
                01450                                   ;THE 'TRANSFER' ADDRESS
72D9 0A0A       01460         DEFW    0A0AH           ;TWO LINE FEEDS
72DB 57         01470         DEFM    'WHEN READY, HIT SPACE BAR TO BOOT'
    48 45 4E 20 52 45 41 44 59 2C 20 48 49 54 20 53
    50 41 43 45 20 42 41 52 20 54 4F 20 42 4F 4F 54
72FC 80         01480         DEFB    80H             ;EDTASM TERMINATOR
                01490 ;
                01500 ;ENTRY POINT AFTER EDTASM RELOCATED BY ROUTINE @ 7330H
                01510 ;
72FD            01520 BEGIN   EQU     $
72FD 318043     01530         LD      SP,4380H        ;EDTASM STACK
```

```
7300 2A1082   01540         LD    HL,(BFSTAD)     ;START OF EDTASM TXT BUF
7303 223442   01550         LD    (4234H),HL      ;STORE IN EDTASM
7306 E5       01560         PUSH  HL              ;START OF EDTASM TXT
7307 2A1282   01570         LD    HL,(BFEND)      ;END OF EDTASM TXT BUF
730A 223242   01580         LD    (4232H),HL      ;STORE IN EDTASM
730D 23       01590         INC   HL
730E 23       01600         INC   HL
730F 00       01610         NOP
7310 D1       01620         POP   DE              ;START OF EDTASM TEXT BUF
7311 AF       01630         XOR   A               ;CLEAR CARRY
7312 ED52     01640         SBC   HL,DE           ;LENGTH TO MOVE
7314 44       01650         LD    B,H
7315 4D       01660         LD    C,L             ;BC HAS BYTE COUNT
7316 211482   01670         LD    HL,BUFFER       ;START OF MOVE
7319 EDB0     01680         LDIR                  ;MOVE DUMPED TXT TO EA+
731B 21FFFF   01690         LD    HL,MEMEND
731E 223642   01700         LD    (4236H),HL      ;STORE IN EDTASM
7321 2B       01710         DEC   HL
7322 223442   01720         LD    (4234H),HL      ;ANOTHER EDTASM PTR
7325 C33943   01730         JP    4339H           ;EDTASM WARM START ENTRY
              01740 ;
              01750 ; THIS SEQUENCE RELOCATES EDTASM+ AFTER LOADING FROM DISK
              01760 ;
7330          01770         ORG   7330H
7330 F3       01780         DI
7331 011638   01790         LD    BC,ENDTXT-4380H+2       ;# BYTES TO MOVE
7334 210052   01800         LD    HL,5200H        ;HL => SOURCE
7337 118043   01810         LD    DE,4380H        ;DE => DESTINATION
733A EDB0     01820         LDIR                  ;MOVE DUMPED PROGRAM
733C C3FD72   01830         JP    BEGIN           ;GO TO NEXT STEP
              01840 ;
              01850 ;++++++++++++++++++++++++++++++++++++++++++++++++++
              01860 ;+                                                +
              01870 ;+THIS IS THE END OF THE ORIGINAL PATCH WHICH APPEARED +
              01880 ;+IN THE REFERENCED '80 MICROCOMPUTING' ARTICLE.      +
              01890 ;+                                                +
              01900 ;+THE FOLLOWING PATCHES ARE THE AUTHOR'S TO ALLOW USE  +
              01910 ;+OF THE PREVIOUS MATERIAL ON A MODEL III.            +
              01920 ;+                                                +
              01930 ;++++++++++++++++++++++++++++++++++++++++++++++++++
              01940 ;
              01950 ;     ++++++++++++++++++++++++++++++++++++++++++++
              01960 ;     AFTER EDTASM+ IS MOVED INTO PLACE, CHANGE 'CALL
              01970 ;     4409H' AT 4FD4H & 4FFEH TO CALL HERE INSTEAD
              01980 ;     ++++++++++++++++++++++++++++++++++++++++++++
              01990 ;
733F          02000 CASSON  EQU   $               ;PATCH WHEN TAPE STARTS
              02010 ;
733F CD0944   02020         CALL  4409H           ;EA+ CALL WE INTERRUPTED
              02030                               ;TO GET 'READY CASSETTE'
              02040                               ;MESSAGE
7342 AF       02050         XOR   A
7343 321142   02060         LD    (4211H),A       ;SET LOW CASSETTE SPEED
7346 210342   02070         LD    HL,4203H        ;LOCN OF 'BK' VECTOR
              02080                               ;DURING TAPE LOADS
7349 118073   02090         LD    DE,SAVBRK       ;PLACE TO STORE IT
734C 010300   02100         LD    BC,03H          ;# CHARS TO MOVE
734F EDB0     02110         LDIR                  ;SAVE 'EM
7351 3EC3     02120         LD    A,0C3H          ;'JUMP' INSTRUCTION
7353 320342   02130         LD    (4203H),A       ;FOR TAPE 'BREAK' TO GO
7356 218342   02140         LD    HL,4283H        ;TO EA+ 'WARM' REENTRY
7359 220442   02150         LD    (4204H),HL      ;PUT THIS IN JUMP VECTOR
735C 1ED3     02160         LD    E,0D3H          ;RELOAD E WITH SOURCE
              02170                               ;TAPE'S FIRST BYTE
735E C9       02180         RET                   ;TO EA+ MAIN PROGRAM
              02190 ;
              02200 ;     ++++++++++++++++++++++++++++++++++++++++++++
              02210 ;     AFTER EDTASM+ IS MOVED INTO PLACE, CHANGE 'CALL
              02220 ;     01F8H' AT 44CFH, 4D1FH, 6580H & 71C6H TO CALL
              02230 ;     HERE INSTEAD.
              02240 ;     ++++++++++++++++++++++++++++++++++++++++++++
              02250 ;
735F          02260 CASOFF  EQU   $
              02270 ;
735F 3A1342   02280         LD    A,(4213H)       ;GET EA+ BYTE
7362 328373   02290         LD    (STORE),A       ;SAVE IT
7365 3E04     02300         LD    A,04H           ;'REGULAR' DOS BYTE
7367 321342   02310         LD    (4213H),A       ;PUT IT THERE
736A CDF801   02320         CALL  01F8H           ;TURN OFF CASSETTE
736D F3       02330         DI                    ;A 'M U S T'
736E 3A8373   02340         LD    A,(STORE)       ;GET EA+ BYTE BACK .
7371 321342   02350         LD    (4213H),A       ;AND REPLACE IT
7374 218073   02360         LD    HL,SAVBRK       ;RESTORE 'BK' VECTOR
7377 110342   02370         LD    DE,4203H
737A 010300   02380         LD    BC,0003H
737D EDB0     02390         LDIR
737F C9       02400         RET                   ;TO EA+
0003          02410 SAVBRK  DEFS  3               ;STORAGE FOR 'BK' VECTOR
7383 00       02420 STORE   DEFB  00H             ;SAVE (4213H) HERE
000C          02430 EXTRA   DEFS  0CH             ;TO START BUFFER AT 7390H
              02440                               ;AND, IN CASE I THINK OF
              02450                               ;ANYTHING ELSE!
              02460 ;
              02470 ;WE MUST SET UP DUMMY TEXT AND ADDRESSES IN EVENT WE
              02480 ;START WITH AN EMPTY TEXT BUFFER. THIS IS REQD SINCE
              02490 ;START-UP IS THROUGH THE 'WARM' ENTRY TO EDTASM+. IF A
              02500 ;REAL SOURCE IS USED, THE 'DUMMY' WILL BE OVERLAID.
              02510 ;
7390 9073     02520 BUFST   DEFW  $                ;EA+ WILL START SOURCE
              02530                               ;TEXT AT THIS ADDRESS
7392 9073     02540 BUFEND  DEFW  $-2             ;SEE 10/81 '80 MICRO' ART
7394 FFFF     02550 ENDTXT  DEFW  0FFFFH          ;EA+ END-OF-TEXT MARKER
0000          02560         END
00000 TOTAL ERRORS

BEGIN  72FD   BFENAD 8212   BFSTAD 8210   BUFEND 7392   BUFFER 8214
BUFST  7390   CASOFF 735F   CASSON 733F   CNVL   7284   CNVR   7288
DMPADR 72CE   ENDTXT 7394   EXTRA  7384   HEXOK  7292   MEMEND FFFF
SAVBRK 7380   STORE  7383   TEXT   7295   WTSPC  64C3
```

MODEL III BASIC PATCH (contributed by Nate Salsbury) - Here's a patch for the BASIC/CMD program of TRSDOS 1.3. This patch corrects the problem that if you scroll a program listing using the up-arrow or down-arrow keys, you'll "lose" two bytes of MEM for each hit of a key. More important, if you put a STOP in a program to debug it, and then scroll the listing with those keys, then when you try to CONTinue you'll get a "Can't Continue" or "Next Without For", etc. error message if this patch is not installed! The patch is:

PATCH BASIC/CMD (ADD=58C4,FIND=D5,CHG=00)

The problem that this patch corrects is due to improper stack operation, and is caused by an earlier "official" Tandy patch, which eliminated two bytes being ADDED to MEM when the SHIFT-up-arrow key was depressed (a bug that also appeared in early versions of NEWDOS, even before the Model III came onto the market, but that's another story). The Tandy patch (which should already be applied to your BASIC/CMD - if not, do this one first) was as follows:

PATCH BASIC/CMD (ADD=58F8,FIND=F1,CHG=00)

Nate says he got the above patch from two separate people, both of whom said they had used it for quite a while with no ill effects, and he reports that he's personally had no problems with it after using it for a month.

CREATE A SELF-BOOTING DISKETTE USING NEWDOS/80 - I recently received a letter from Joachim Kelterbaum of Essen, West Germany in which he describes the process. Since Joachim has a Model I, I don't know if this would work on a Model III or not. However, this is the procedure he used to create a self-booting disk for the Model I:

Format a data disk (no system on it) - note that the TSR (Track Step Rate) for boot must be set properly by PDRIVE before formatting the data disk. Now copy the /CMD file you want to be self-booting to the formatted data disk. Use the DIR,d,A command (where d is the drive number of your data disk) to make sure that your file was saved in one contiguous block (it must have only one extent, as listed in the EXTS column of the directory display). Now use SUPERZAP's DFS-option to find out starting track and sector in that track of this particular file. Then patch relative bytes 12 and 13 of BOOT/SYS of that diskette the following way:

12 = start sector # of your /CMD file in that track.

13 = start track # of /CMD file (absolute value e.g. track # - 1 if double density because of the single density track 0).

Also patch relative bytes 47 and 48 (now contain a CP 0A5H instruction) with two NOP (00H) bytes. Your self-booting diskette should now be working.

If you now destroy some unused tracks by over-formatting them in a different density or something similar, then your diskette will even be 'protected' (which, Joe observes, will probably teach potential pirates to get more sneaky).

BACKING UP MORE RECENT (3.0 ON) VERSIONS OF SUPER UTILITY PLUS by Paul Snively - This particular technique is original with me, but I had some local help. Kudos to: J. Joseph Felten, for being a good Z-80, and to Marshall Dunbar, for being a good stack and register set. Yup, you guessed it - we cracked SU+ 3.1a by disassembling the bootstrap and "playing computer" with it (I was the FDC. Boy, was that fun!!))

I won't go into all the gory details. Suffice it to say that the following method works: First, boot SU+ (we used 3.1a, which was current at the time.) Next, boot a disk with your favorite monitor (which, incidentally, should be TASMON. If it isn't, switch monitors!!)) USE THE RESET KEY TO RE-BOOT, NOT SU+'s EXIT OPTION! Go into the monitor and put these bytes in a convenient RAM spot: ED 57. Put a breakpoint after this instruction. With TASMON, this would be done via the M command. Example:

M H 6000 ED 57 <BREAK> B 1 6002

Having done that, <G>o to 6000 (or wherever you put the bytes. You should now be back in the monitor (thanks to the breakpoint.) Look at the A register. Write down the value you find there. THIS IS A CHECKSUM OF THE SERIAL NUMBER OF YOUR SU+!!! Believe me, you'll need to know this value.

OK. Now boot SU+ again. Using the MOVE MEMORY option in the MEMORY UTILITIES menu, move memory from 4000H - 67FFH to D500H. This will put the lower 10K of SU+ well out of the way of DOS and its library routines. Next, at FD00H, enter the following bytes:

F3,21,00,D5,11,00,40,01,00,28,ED,B0,3E,XX,ED,47,
31,00,FE,C3,15,40

XX, of course, is the value that you wrote down. When you have done this, boot your favorite DOS (again, with the reset button) and use the appropriate DUMP syntax for your DOS with the start address being 6800H, end address is FD15H, and executing address is FD00H. Congratulations. You now have SU+ as a /CMD file.

Now then. What good is SU+ as a /CMD file? That's a good one. First of all, it allows SU+ to be backed up very easily. Secondly, if there are any patches that you want to apply, it's much easier to do it to the /CMD file than to the original disk. Lastly, you can disassemble SU+ (also lots of fun,) or run it under your monitor.

What? The /CMD file overwrites your monitor in high memory? I know the feeling. Some of you will be more or less out of luck, but you MULTIDOS and Newdos/80 users (you always !/knew!? that you have superior DOSes, didn't you?) can bypass the problem easily. Here's how:

Get your monitor in memory. I use TASMON 2.12, which is relocatable, but I tend to put it from E000H to FFFFH. Once your monitor is in, go back to DOS and type

DUMP MON/CIM:0 (START=X'SSSS',END=X'EEEE',CIM)

SSSS and EEEE are, of course, the starting and ending addresses of the monitor. What's this weird syntax? This syntax, which exists only in Newdos/80 and MULTIDOS (thanks, Apparat and Vern) puts the file on disk WITHOUT THOSE NASTY LOADER CODES! That's right; it simply puts the bytes in memory straight out to disk.

"So?" So, if you configure your SU+ for MULTIDOS and locate the file on the disk (using SU+'s FILE LOCATIONS option,) you can then simply READ THE FILE SECTORS TO THE APPROPRIATE HIGH MEMORY LOCATIONS AFTER SU+ IS ALREADY RUNNING!

Wow. Now that you have your monitor in memory, though, there is one thing that you need to do before you jump to it: apply the following patch to SU+ (in memory):

Modify memory at A8CEH. Zap all bytes up to but not including the E9H to zeroes. Now go to the memory utilities and Jump to your monitor. Fine. To get back to SU+, <G>o to 4015H. Viola!

*** NOTICE *** The above patch works on SU+ 3.1a assembled on March 16, 1983. It may or may not work on other versions. I have no other versions, so don't come crying to me to develop the patch for you. If you're desperate, however, you can send me a copy of your /CMD file backup, and I will find the patch point and a) send you a patch sheet, or b) call you (collect. Sorry about that) and tell you what to patch and where. I will then kill the /CMD file that you sent me. Jesse Bob Overholt (of TAS) says that getting into the SU+ patch business is a mistake. I'm not so sure, but if I find that I just can't keep up, your requests may simply go unanswered. If you want to discuss SU+ (good points, bad point, idiosyncracies, patches, protection scheme (yes, I'm quite well versed on it),) then call me or write me. My name is Paul Snively and I live (for the moment, anyway) at Ashton-Hershey 12, Indiana University, Bloomington, Indiana 47405 - telephone (812) 337-1570. I can't accept collect calls on the university phone system, so don't bother.

MODEL 100 ASSEMBLY LANGUAGE FILE TRANSFER PROGRAM by Gordon MacSwain - This program should be of considerable interest to those of you that own (or have access to) both a Model 100 and a Model I/III/4, and that program in assembly language. As is becoming typical in NORTHERN BYTES, the documentation (such as it is) is in remark statements within the program itself. Note that Gordon's phone number and address (except for his postal code, which is M1T 2B5) are in lines 7-9, in case you have questions or comments about the program. Here's the listing, all ready for you to type into your Model 100:

```
0 ' ***** COCOPY.BA ***** Feb 25/84
1 ' 80 MICRO (Jan '84, P171) has a COM/CMD program that enables
2 ' transfer of TRS80 Model 100 files, to or from Models 1/3/4
3 ' for disk storage.
4 ' Unfortunately, the Model 100 ROM does not provide for the
5 ' saving or loading of ".CO" (assembly lang.) files via the
6 ' com port, as it does for ".BA" & ".DO". "COCOPY.BA" gives
7 ' you that capability.          MacSwain Enterprises
8 '                              151 Cass Ave, Agincourt, Ontario
9 '                              (416) 291-7673      <72155,1461>
10 CLS:PRINT:PRINTTAB(10);CHR$(27)+"p";"***** COCOPY *****" ;CHR$(27)+"q"
20 PRINTTAB(10);CHR$(27)+"p";"macswain enterprises";CHR$(27)+"q":PRINT:LINE(57,5)-(181,25),1,B:LINE(60,7)-(179,7):LINE(59,7)-(59,23)
30 PRINTTAB(8);"<S>ave or <L>oad CO File";
40 K$=INKEY$:IFK$="L"ORK$="l"THEN180ELSEIFK$="S"ORK$="s"THEN50ELSEIFK$=CHR$(27)THENMENUELSE40
50 Z=63930:PRINT:PRINTTAB(3);"On Model 1/3 Run: 'COM I FILENAME'":INPUT"Then Enter CO Filename (No Ext)";A$
60 FORX=0TO198STEP11
70 IF(PEEK(X+Z)AND160)=160THENGOSUB100
80 NEXTX
90 PRINTTAB(13);"FILE NOT FOUND";:FORW=1TO500:NEXTW:GOTO10
100 B$="":A=LEN(A$):FORY=1TOA:B$=B$+CHR$(PEEK(X+Z+2+Y)):NEXTY
110 IFA$<>B$THENRETURN
120 IFA<6ANDPEEK(X+Z+2+Y)<>32THENRETURN
130 B=(PEEK(X+Z+2)*256)+PEEK(X+Z+1)
140 H$="0123456789ABCDEF":OPEN"COM:98E1E"FOROUTPUTAS1
150 X=6+(PEEK(B+3)*256)+PEEK(B+2)
160 FORY=0TOX-1:G=G+1:C$="":C$=C$+MID$(H$,((PEEK(B+Y)AND240)/16)+1,1)+MID$(H$,(PEEK(B+Y)AND15)+1,1):PRINT#1,C$;:IFG=16THENPRINT#1,",";:G=0:NEXTY:ELSENEXTY
170 PRINT#1,CHR$(26):CLOSE1:PRINTTAB(15);"FILE SAVED";:FORW=1TO500:NEXT:GOTO10
180 PRINT:PRINTTAB(3);"On Model 1/3 Run: 'COM O FILENAME'"
190 OPEN"COM:98E1E"FORINPUTAS1
200 INPUT#1,B$
210 H$="BCDEF":R=LEN(B$)
220 X=1:GOSUB300:LD=A*16+B:X=3:GOSUB300:LD=((A*16+B)*256)+LD:PRINT"START ";LD
230 X=5:GOSUB300:LN=A*16+B:X=7:GOSUB300:LN=((A*16+B)*256)+LN:PRINT"LENGTH";LN
240 X=9:GOSUB300:ST=A*16+B:X=11:GOSUB300:ST=((A*16+B)*256)+ST:PRINT"ENTRY ";ST
250 INPUT"Create CO Filename (No Ext)";F$
260 Z=LD:FORX=13TORSTEP2:GOSUB300:POKEZ,A*16+B:Z=Z+1:NEXTX
270 INPUT#1,B$:R=LEN(B$):FORX=1TORSTEP2:GOSUB300:POKEZ,A*16+B:Z=Z+1:NEXTX
280 IFEOF(1)THEN290ELSE270
290 PRINTTAB(14);"FILE CREATED":SAVEMF$,LD,LD+LN-1,ST:END
300 IFMID$(B$,X,1)>"9"THENA=10+INSTR(H$,MID$(B$,X,1))ELSEA=VAL(MID$(B$,X,1))
310 IFMID$(B$,X+1,1)>"9"THENB=10+INSTR(H$,MID$(B$,X+1,1)):RETURNELSEB=VAL(MID$(B$,X+1,1)):RETURN
```

DOSPLUS HINTS by Dave Bower - I have a couple of suggestions for the DosPlus user who has one or more of his drives configured as single sided.

A single VOLUME drive is a double-headed drive addressed as one drive. For example, you can run four double-headed drives on a Model III, if they are configured as single volume, each set of heads is ONE drive.

This is great, unless you want to back up a single VOLUME diskette to a single SIDED diskette. This can be a real pain in the neck! Copying each file, keeping track on paper of what you copy. Whether you copy file by file, or use Super Utility you still have to sit there and manually keep track of what copies until you fill each single SIDED diskette. Unless you let DosPlus work for you.

Using the KILL parameter and the wildmask copy DosPlus will copy all the files, killing each one as it copies. Once you fill up a diskette put a fresh one in and keep copying till you run out of files to copy.

1) Create a backup of the original diskette. Work with the backup, in case anything goes wrong. You'll need two or three single SIDED data diskettes.

2) Be sure your configuration table is set up properly for the drives containing the different diskettes.

3) Use a wildmask copy and specify the KILL and the ECHO parameters. Here we are copying from drive :1 (the single VOLUME) to drive :2 (the single SIDED).

COPY :1 :2 (ECHO,KILL)

4) DosPlus will copy files until it runs out of room. Kill the last file echoed on the screen, because it was not copied completely. Switch data diskettes and once again:

COPY :1 :2 (KILL,ECHO)

If necessary do this again until all the files are copied to single sided diskettes. I have found that it may take as many as 3 single sided diskettes to contain all the files on one double sided diskette.

My second suggestion also concerns single and double sided diskettes. The quickest way to trash a diskette beyond repair is have the SIDES parameter set wrong. Normally I have a SYSTEM file to take care of this. BUT just in case I forget, or it doesn't get loaded I have the cursor on all my unconfigured diskettes set to ASCII 196 (the smiley face on the Model III). If I see the smiley face I know that I haven't loaded a SYSTEM file and that only my first drive (if even that one) is set for SIDES=2.

WHY ALIGN ?? by Dave Bower - There are three times in the life of a disk drive when an alignment should be performed. One, is when it is brand new and before diskette one is run through it; two, is the FIRST time errors appear during disk I/O and three is at 6 month intervals.

The most overlooked drive alignment is the first one by the new owner. It is generally assumed (often incorrectly) that a new drive is in top condition and aligned to perfection. The initial alignment will accomplish three things. It ensures that the drive is in working order. It can also flush out any intermittent problems lurking inside that will become serious the day after the warranty expires. Secondly, it will ensure that your drive is properly aligned, and if your drives are aligned by the same service, you will have no problem with compatibility with other drives in your system. Third, and most important, it sets a standard by which all your diskettes will be written. If, on the other hand, you put off your first alignment until CRC errors become unbearable you will probably have trouble reading diskettes previously made by the drive.

Unscheduled alignments should be made as soon as you suspect ANY alignment problem whatsoever. Any drive will usually read what it writes no matter what the state of alignment, that is until it is properly aligned again, and then all the data laid down by the misaligned drive is lost. If you put off alignment of a drive until the last possible minute then first back-up all your diskettes made with the bad drive to a known good drive.

After you've brought your drives up to the proper state of performance the next step is to keep them there. You do this by having the drive aligned twice a year. This not only provides the obvious, but can prevent or highlight problems before they become serious. A thorough cleaning will help protect your head, mechanical parts and diskettes. The repair service will be aware of any service changes or modifications put out by the manufacturer of your drive. If you use the same repair service he will have a history file of your individual drives and can use this to spot potential problem areas. And a history of preventive maintenance will be helpful should you decide to sell your drives.

In the end you are going to have to decide for yourself what kind of maintenance schedule you need. Just remember the difference between scheduled and unscheduled maintenance; scheduled maintenance is performed at your convenience, unscheduled maintenance is performed at the worst possible time.

GREMLINS ON THE POWER LINE by Dave Bower - I used to have a buddy who wore a lucky medallion that he swore kept gremlins away. Now we were about 7 years old and didn't like monsters, but he was especially terrified of gremlins. I don't think I ever saw him without his lucky medallion. I used to think he was crazy. But, come to think of it, I never saw any gremlins when he was around. Maybe it worked!

I guess we've all got our fears. I'm afraid of voltage surges. That's why I've got my lucky Radio Shack power filter hooked up to my computer at all times. Is there any noticeable difference? I dunno. I've never run the computer without it. But I haven't seen any surges (or gremlins) since I've had it.

What's a surge? I dunno. I've never seen one. That is, until recently. I haven't seen one, but I have learned what one is.

Actually the problem most computerists fear is transients on their power lines. These are very fast transverse-mode (between neutral and hot) changes in the voltage, called spikes and surges.

A SPIKE is defined as a sudden disturbance of 6,000 volts or greater lasting less than 100 microseconds. In some case spikes can cause transistors to fail and erasure of solid state memory. Components continually subjected to spikes will suffer from a shortened life span. A SURGE is a burst of 3,000 volts lasting longer than 100 microseconds.

You can expect more than 100 such spikes and surges each month. And you will probably get a surge of at least 1,000 volts at least once a day. In fact, spikes and surges account of 88.5% of all disturbances.

Like I said, I've never had any problems with power-line fluctuations, but I've never run my system without a power-line filter. If you're having problems with your system, this might be a good place to start looking.

(The information for this article was taken from "Power Line Protection", an article by Douglas Pryor and Amy Smith in the December 1983 BUSINESS COMPUTER SYSTEMS)

THE WORLD'S MOST EXPENSIVE SYNTAX ERROR is excerpted from an article by Bob Gardner, that originally appeared in BUSS-LINE (newsletter of the Central Florida Computer Society, Inc.)!

No, I didn't make the world's most expensive syntax error, but I learned what it was recently when I attended the Air Force conference on Technology in Training and Education at Sheppard AFB in beautiful downtown Wichita Falls, Texas. During a tutorial session on ADA (ADAptability, reADAbility) given by Major Richard Bolz, formerly of the Air Force Academy, we who claimed to have programmed FORTRAN were asked to explain how the following code fragment would work!

```
      DO 10 I=1.3
      ~~~~~~~~~~
      ~~~~~~~~~~
      ~~~~~~~~~~
10    CONTINUE
```

We said it would execute the code body three times. But noooooo... that's 1,3 - not 1.3! So somebody said it might truncate 1.3 to 1 and execute the code body once; but that's not it either.

It assigns 1.3 to a weird variable called DO10I (FORTRAN ignores spaces), executes the code body once, ignores the spurious CONTINUE statement, and continues merrily on. Major Bolz said he had been accused of inventing this example to reinforce the case for strongly typed languages like ADA, but he said that this example was the syntax error that lost one of the Venus probes a few years back!

COLOUR COMPUTER WOES is reprinted from the Adelaide (Australia) Micro-User News!

The limited number of colours available in the high resolution mode has prompted American software writers to increasingly use 'false' colours that can be achieved with the American NTSC TV system. These same interference effects do not occur with the European (& Australian) PAL system. This means those extra colours turn into a blur of grey lines. If Tandy wants to lose customers to Commodore they will continue to demonstrate the likes of CANYON CLIMBER (b.... awful!!). If this is a trend for future software and no PAL versions appear then it's a gift to Commodore and Dickie.

ZAPDEBUG - (c) 1983 Tony Domigan, P.O. Box 150, Thomastown, Victoria, Australia, 3074.

ZAPDEBUG evolved from the need to modify TRSDOS 1.3B sectors when there was no satisfactory utility available. The program patches the RST function so that DEBUG can be loaded without executing. Once loaded, DEBUG's 'F' command is patched, the RST function is fixed and DEBUG is executed.

When the 'F'ilespec option is selected, if a drivespec is entered in place of a filespec then ZAPDEBUG takes over. To use ZAPDEBUG, therefore, you need to enter 'F' and answer the Filespec prompt with the drivespec of your choice e.g. '0'. You will then be prompted for the sector number you require (in Hexadecimal) e.g. 132, and it will be read into the buffer at 4300H. Once in the buffer you can modify the sector as in normal DEBUG, and when the ENTER key is hit the buffer will be written to the disk.

Hitting the BREAK key returns you to the FILESPEC prompt. You can then enter a drivespec or filespec or if you wish, hit the BREAK key again and you will return to normal DEBUG (in place of DOS).

[The source code for ZAPDEBUG follows:]

```
                00001 ;              ZAPDEBUG
                00002 ;       (c) 1983 Tony Domigan
                00003 ;       PO Box 150, Thomastown,
                00004 ;       Victoria, Australia,3074.
                00005 ;
7000            00006          ORG     7000H
7000 210E70     00007 START    LD      HL,RSTART       ;My Patch Address
7003 3EC3       00008          LD      A,0C3H          ;= JP
7005 32CD40     00009          LD      (40CDH),A       ;Patch RST with JP
7008 22CE40     00010          LD      (40CEH),HL      ;to RSTART
7008 3E05       00011          LD      A,05H           ;= Overlay 5 (DEBUG)
700D EF         00012          RST     28H             ;Load DEBUG
700E 3ECD       00013 RSTART   LD      A,0CDH          ;= CALL
7010 212C70     00014          LD      HL,MYCK         ;My pre-processing
7013 32E052     00015          LD      (52E0H),A       ;Patch 'Open Filespec'
7016 22E152     00016          LD      (52E1H),HL      ;Route to my check
7019 3E22       00017          LD      A,22H           ;Re-patch RST function
701B 21074B     00018          LD      HL,4B07H
701E 32CD40     00019          LD      (40CDH),A
7021 22CE40     00020          LD      (40CEH),HL      ;RST now Normal
7024 3EE2       00021 FDORET   LD      A,0E2H          ;Patch JP2DOS with..
7026 32D752     00022          LD      (52D7H),A       ;JP to Zapdebug
7029 C3004E     00023          JP      4E00H           ;Execute DEBUG
                00024 ;
702C 1A         00025 MYCK     LD      A,(DE)          ;DE ==> Filespec
702D FE34       00026          CP      34H             ;Ck for Drives 0 to 3
702F 3804       00027          JR      C,YES           ;Jump if Yes
7031 CD2444     00028          CALL    4424H           ;Zap not reqd - Open File
7034 C9         00029          RET                     ;DEBUG take over
                00030 ;
7035 E5         00031 YES      PUSH    HL
7036 D630       00032          SUB     30H             ;Convert Drive to Binary
7038 328A70     00033          LD      (FCB6),A        ;Post Drive No. to FCB
703B 215C70     00034          LD      HL,MSG          ;Point to Sector Prompt
703E CD1B02     00035          CALL    021BH           ;Display Msg
7041 CD1F52     00036          CALL    521FH           ;DEBUG Hex Input Rtne
7044 7D         00037          LD      A,L
7045 328E70     00038          LD      (FCB10),A       ;LSB of Sector to FCB
7048 7C         00039          LD      A,H
7049 328F70     00040          LD      (FCB10+1),A     ;MSB of Sector to FCB
704C 218A70     00041 MVFCB    LD      HL,FCB          ;Source = My FCB
704F 115555     00042          LD      DE,5555H        ;Dest.  = DEBUG FCB
7052 012F00     00043          LD      BC,002FH        ;Length of FCB
7055 EDB0       00044          LDIR                    ;Move it all
                00045 ;
7057 E1         00046 GODOIT   POP     HL
7058 AF         00047          XOR     A               ;Sets Z Flag (no error)
7059 C31253     00048          JP      5312H           ;Let Debug Take Over
705C 0A         00049 MSG      DEFB    0AH
705D 45         00050          DEFM    'Enter Sector Number to Zap (in Hex) : '
    6E 74 65 72 20 53 65 63 74 6F 72 20 4E 75 6D 62
    65 72 20 74 6F 20 5A 61 70 20 28 69 6E 20 48 65
    78 29 20 3A 20
7083 03         00051          DEFB    03H
7084 80         00052 FCB      DEFB    80H             ;Read-Only file is open
7085 00         00053 FCB1     DEFB    00H             ;Full Sector I/O
7086 00         00054          DEFB    00H
7087 0043       00055 FCB3     DEFW    4300H           ;256 byte buffer
7089 00         00056          DEFB    00H
708A 00         00057 FCB6     DEFB    00H             ;Drive Number
708B 0000       00058          DEFW    0000H
708D 00         00059          DEFB    00H
708E 0000       00060 FCB10    DEFW    0000H           ;Start Relative sector No
7090 D002       00061 FCB12    DEFW    02D0H           ;Max relative sector=720D
7092 0000       00062          DEFW    0000H
7094 001E       00063 FCB16    DEFW    1E00H           ;01 = First Extent
7096 051E       00064 FCB18    DEFW    1E05H           ;02
7098 091E       00065 FCB20    DEFW    1E09H           ;03
709A 0F1E       00066 FCB22    DEFW    1E0FH           ;04
709C 141E       00067 FCB24    DEFW    1E14H           ;05
709E 191E       00068 FCB26    DEFW    1E19H           ;06
70A0 1E1E       00069 FCB28    DEFW    1E1EH           ;07
70A2 231E       00070 FCB30    DEFW    1E23H           ;08
70A4 FFFF       00071 FCB32    DEFW    0FFFFH          ;10
70A6 FFFF       00072 FCB34    DEFW    0FFFFH          ;11
70A8 FFFF       00073 FCB36    DEFW    0FFFFH          ;12
70AA FFFF       00074 FCB38    DEFW    0FFFFH          ;13
70AC FFFF       00075 FCB40    DEFW    0FFFFH          ;14
7000            00076          END     START
00000 TOTAL ERRORS

FCB     7084    FCB1    7085    FCB10   708E    FCB12   7090    FCB16   7094
FCB18   7096    FCB20   7098    FCB22   709A    FCB24   709C    FCB26   709E
FCB28   70A0    FCB3    7087    FCB30   70A2    FCB32   70A4    FCB34   70A6
FCB36   70A8    FCB38   70AA    FCB40   70AC    FCB6    708A    FDORET  7024
GODOIT  7057    MSG     705C    MVFCB   704C    MYCK    702C    RSTART  700E
START   7000    YES     7035
```

BOB BRUMLEY'S HACKER HINTS - These were gleaned from a phone conversation with Bob Brumley at Software Success in Windsor, California:

Would you like your Model I video display to be as sharp as Model 4 display? Bob says that a Model 4 video board can be used in the Model I monitor. You do have to get the horizontal and vertical sync signals from the Model I keyboard unit to the monitor, but this is made easier by the fact that the five pin DIN connector used for video output has a couple of unused pins that can be appropriated for the purpose. In addition, there's a blank space on the Model 4 video board that is large enough to be drilled out for an LM383 audio amplifier IC, so if you add one of those and a speaker inside the monitor cabinet, you can have built-in sound.

You say that one's beyond your capabilities? Well, here's another. Perhaps you've wanted to add the reverse video circuit from Dennis Kitsz's book, "The Custom TRS-80", to your Model I, but you have a Holmes Engineering SPRINTER installed and they both use port 254 (0FEH) for control. The actual conflict is at data line 1 of port 254, which is already accessed by the SPRINTER. No problem, just use data line 4 instead of line 1. To do this, move the reverse video connection from Z59 pin 5 to Z60 pin 2 (or any other convenient connection to data line 4). Now port 254 can be used to set both the SPRINTER and the reverse video at the same time - just use the normal values to set SPRINTER speed and normal video, or the normal SPRINTER values + 16 to set SPRINTER speed and reverse video. By the way, Bob notes that your memory must be rated at 150 ns or better (120 ns is preferable) to run the SPRINTER at 5.3 MHz without unexpected crashes.

Apparently, the program stored in the PAL chip in the Model 4 defines the maximum amount of memory that can be used in that Model. Bob says he heard of one guy that programmed his own PAL, and now has a Model 4 with 512K of memory!

Bob is working on a book entitled "NEWDOS/80 Decoded" (it will have a format similar to "Microsoft BASIC Decoded", including commented disassemblies of the NEWDOS/80 /SYS modules), but in addition, he will show you how to add some additional features to that DOS. If you've ever said, "I like NEWDOS/80, but I wish it had this feature...", you might drop a line to Bob and let him know of your interest (send us a copy here at NORTHERN BYTES, maybe we'll build our own "wish list"). Perhaps Bob will figure out a way to add the DOS command of your dreams. Write to Bob Brumley, Software Success, 8522 Alden Lane, Windsor, California 95492.

MODEL 100 HINT (as reported by Ken Ashley of OCTUG in ORANGE BYTES):

When uploading Model 100 document files to another computer, precede the text with an 'at' sign (@). This will prevent the first three characters from being lost upon transmission. The @ is not transferred.

DETERMINING THE DOUBLER IN USE ON A TRS-80 MODEL I by Paul Snively - First, there are only two kinds of doublers: Radio Shack's, and everybody else's. So, with that thought in mind, here is a subroutine that determines what kind of doubler a Model I has:

```
LD      IY,37ECH
LD      DE,80A0H
LD      HL,(37EDH)
LD      (IY+2),D
LD      (IY+1),D
LD      (IY+2),E
LD      (IY+1),1
LD      (IY+2),D
LD      A,(IY+1)
LD      (IY+2),E
LD      (37EDH),HL
DEC     A
RET
```

If the zero flag is set upon returning from this routine, the doubler is a non-Radio Shack (Percom, Aerocomp, etc.), otherwise it's a Radio Shack doubler. Neat, huh?

Actually, there's one hitch. If the zero flag is set, it's either a non Shack doubler or else there's NO DOUBLER AT ALL! So, if it's necessary to know if there's even a doubler, it'll take more code. If you need the extra code, let me know and I'll write it for you (I've never needed it, so I've never written it).

MULTIDOS 1.6 PATCHES provided by Paul Snively - If you're still using version 1.6 for the Model I (as opposed to version 1.6a, which fixes this problem), here's a patch that eliminates Vern's keyboard debounce routine (and also eliminates "lost" keystrokes):

If you're using single density, the track/sector is 1/6. If you're using double, the track/sector is 1/8. If you're using P density, the track/sector is 1/14. In any case, use ZAP/CMD (or SU+, etc.) and change relative byte 6E (all densities) from: F1 A6 C8 21 to: F1 00 00 21

There you go. While I'm at it, here are some other 1.6 zaps (all courtesy of the February, 1983 MULTIDOS newsletter!!)

If your ZAP/CMD is version 1.2 or 1.3, then apply the appropriate following zap:

If version 1.2: Relative file sector 2, relative byte 14H change: 20 40 E5 DD E5 CD to: 20 40 D9 DD E5 CD and relative byte 4EH change: E1 E1 E6 5F to: E1 D9 E6 5F

If version 1.3: Relative file sector 2, relative byte 16H change: 20 40 E5 DD E5 CD to: 20 40 D9 DD E5 CD and relative byte 50H change: E1 E1 E6 5F to: E1 D9 E6 5F.

Single density Track/Sector 0/9, Double density 1/1, P density 0/7. Relative byte DEH for all densities. Change 00 B7 20 to 00 AF 20. This zap is optional, and allows for a one second delay on read. This is necessary if your drives have trouble coming up to speed quickly when selected.

Here's a neat one: Single density 0/7, Double density 0/9, P density 0/5. Relative byte: 19H. Change 3E 00 D3 FF to 3E ub D3 up. "up" is a user defined port, and ub is a user defined byte. This allows the user to make MULTIDOS turn certain hardware mods on on boot-up. Vern, of course, uses this feature to turn his high-speed clock on (and so, for that matter, do I.) Neat, huh?

Another good one: All densities, track 17, sector 0. Yes, this is the GAT sector, but as you may have noticed from time to time, there's PLENTY of room for other things besides the GAT information. Well, Vern puts several system configuration parameters there. On 1.6, Vern didn't allow you to choose how many retries MULTIDOS would make on an error before it actually reported the error (this capability has been added to the CONFIG command on 1.6a). So, to set the number of retries, look at relative byte C4H. If the value is a 04, take the number of retries you want, subtract one from it, and put it here. For example, ten retries minus one is nine, so zap a 09H at relative byte C4H. However, if the byte is a 05, simply zap the number of retries you want here. So, ten retries means zap a 0AH into relative byte C4H. Neat, huh? One thing... the error retries value is limited to 5 bits. So, the maximum number of retries is 31 decimal (1FH).

In the weird filenames dept.... Ever wish that you could enter lowercase filenames and KEEP them lowercase? How about GRAPHICS filenames? (Great for security...) Or numbers as the first character in a filename? What about characters like >, ?, @, !, ", #, $, %, &, and <? MULTIDOS can do it! The remaining three bits of relative byte C4H on 17/0 are the key. If you want lowercase left as such, graphics, and the >, ?, and @ symbols to be valid in filenames, set bit 7. If you want to allow numbers as the first character, set bit 6. If you want the symbols !, ", #, $, %, &, and < to be valid in filenames, set bit 5.

MODEL 4P BOOT MODE KEY SELECTION is reprinted from the Marin County TRS-80 Users Group (MCTUG) Newsletter:

Whether the Model 4P boots in the Model III or Model 4 mode depends on the key sequence used during the boot. Pressing one of several keys will select a different mode of operation. The selected keys should be held until the boot sequence is complete. NEVER USE THE <SHIFT>, <CAPS>, OR <CTRL> KEYS DURING A BOOT SEQUENCE.

NO KEYS PRESSED -- Model 4P boots in the Model III mode. The Model III ROM image must be on the disk.

<F1> KEY PRESSED -- Boot from hard disk.

<F2> KEY PRESSED -- Boot from floppy disk only.

<F3> KEY PRESSED -- Boot in the Model III mode.

<L> KEY PRESSED -- Load the ROM image even if it has already been loaded. This mode is used to force the Model 4P to load the ROM image in case it has been damaged inside the Model 4P.

<N> KEY PRESSED -- Indicates to the Model 4P that it is not to boot the ROM image. Use this mode to boot TRSDOS 6.1.

<P> KEY PRESSED -- Prompts user to switch disks after booting the Model III ROM image. This mode can be used to run previous Model III application diskettes that do not contain the ROM image. At the prompt press <BREAK> to run Model III ROM BASIC, press <ENTER> to load and run a Disk Operating System.

[THE FOLLOWING THREE MODES DO NOT APPEAR IN THE MODEL 4P OWNER'S MANUAL]:

<V> KEY PRESSED -- Prints ROM version on screen. Does NOT boot a disk.

<.> KEY PRESSED -- Performs a dynamic RAM test. Runs continuously until RESET pressed.

<SHIFT><BREAK> KEYS PRESSED -- Boot from RS232. Acts like Network III boot ROM.

A GREEN CRT FOR YOUR TRS-80 - Here's a portion of a letter I received recently, that may be of interest to those of us with black-and-white CRT's:

"Down at the local RS Computer Center, the tech has a couple of green screens in stock. I think they are the CRT's that go in the Model 12/16, but the tech says they will work in the Model 2/3. She says (yes, "she") that they cost about $40, and slapping them in wouldn't cost too much more, especially in conjunction with, say, a disk drive alignment and cleaning.

"Now I don't know if this is 'official' or not, or whether it can get her into trouble, or if she can get green CRT's for the Model 4P. But it is something that might be helpful. Just go and see if the local tech has them."

(signed) Merrill Cook

I suspect that Merrill may have meant to say that the 12/16 screens would work in the Models III or 4, since he mentions the 4P, but in any event the local technician should know what will work and what won't. If anyone else knows of another inexpensive way to get a green (or amber) CRT for the Models I, III, or 4 (or 4P), why not write and let us know (after all, for some of us the nearest Radio Shack Computer Center is a couple of hundred miles away!).

TRSDOS 1.3B PATCHES by Tony Domigan - This is a series of short patches to TRSDOS 1.3b to install the 'R'epeat Dos Command function similar to NEWDOS/80 version 2:

```
PATCH #1 (ADD=4E36,FIND=2642013F003600,CHG=8041013F000000)
PATCH #1 (ADD=5191,FIND=4D4153544552,CHG=522020202020)
PATCH #9 (ADD=603E,FIND=CD8E5FFE0D2003,CHG=21804111254201)
PATCH #9 (ADD=6045,FIND=AF1829111060CD,CHG=2500EDB0212542)
PATCH #9 (ADD=604C,FIND=5444C2,CHG=C39942)
```

REAL-TIME CLOCK FOR THE TRS-80 - I had wondered how hard it would be to build one of these. Well, according to Michael Davis (13816 Paseo Zaldivar, San Diego, California 92129), an article on this very subject (under the title, "Fail-Safe, Real-Time Clock for TRS-80) appeared in the December, 1983 issue of Computers & Electronics (formerly Popular Electronics) magazine. The project used an OKI Semiconductor MSM5832RS clock IC. Mike says he believes the article was written for the Model I, but could be easily adapted for use with the Model III (Mike says that will be his next hardware project!).

DEFEATING CALL WAITING - If you use a MODEM for telecommunications, you may have been told that you cannot have the Call Waiting service offered by the phone company. Call Waiting is a service that permits you to receive a call when your line is already busy. If you are talking to someone on the phone, and someone tries to call you, you will be momentarily disconnected from your first conversation and connected to a "beep" tone. The person on the other end will hear a moment of dead silence (as if you had depressed your phone switchhook for about half a second). Note that the Call Waiting tone is NOT superimposed upon the conversation in progress, rather, the existing connection is momentarily interrupted while the Call Waiting tone is online. You then have the option to hang up, in which case the original connection will be broken, your phone will ring, and you can answer the second call, OR you can switch between calls by momentarily depressing the switchhook - one caller will be on "hold", and you can talk to the other.

This service works reasonably well on voice circuits, but when a phone line is used for both voice and data, some problems arise. For one thing, when a MODEM is in use, it is usually not possible to hear the Call Waiting beep tone, thus a second caller would probably never be answered, and might incorrectly conclude that no one is at home. However, in some circumstances, the momentary interruption of the circuit may cause one or both of the MODEMs in use to drop carrier and disconnect, or at very least may garble the data being transmitted. If a commercial service or a Bulletin Board System is being accessed, the momentary carrier loss will almost always cause a disconnect (and if you are paying for the service, you may incur an extra charge because you didn't terminate the call properly).

Unfortunately, many of us can't afford to have a separate phone line for our computer, so the question is, how do we defeat Call Waiting while our computer is using the phone? You may have been told that it's impossible (especially if you asked someone at the phone company), but that's not so. However, you must have one additional "Custom Calling" service - either Call Forwarding or Three Way Calling - for this to work.

Before I describe the methods, I'll just state that the reason they work is that no matter how many "Custom Calling" features you have, your line will only handle a maximum of two calls at once. In other words, if you have two outgoing calls in progress on Three Way Calling, an incoming call will receive a busy signal even though you also have Call Waiting.

The easiest method involves using Call Forwarding, providing you can find a "permanently busy" number in your area. Most exchanges have one or more numbers that ALWAYS return a busy signal (in Michigan, for example, numbers ending in "9999" are often permanently busy). If you have friends in the phone company, they may be able to give you a permanently busy number. All you do is to set up Call Forwarding to forward all your incoming calls to that number. Callers receive a busy signal, and your line is not interrupted by Call Waiting. In some areas, you may be able to Call Forward calls to your own number, which would have the same effect (busy signal) , or (less desirable) to a number that you are absolutely sure will never answer - perhaps a business you know is closed (this would give callers a ringing signal). Or, you could even forward calls to a friend, who could explain to callers that you're using your MODEM, and either take messages or ask callers to phone back later. Once you're through with your MODEM calls, be sure to cancel the Call Forwarding!

If you have Three Way Calling, you can place a call to a "permanently busy" or "permanently no answer" number, then press the switchhook momentarily to get the second dial tone, and then dial the number you want to call with your computer. The problem with this is that some phone companies have "timeouts" that will automatically disconnect a line that has been connected to a busy signal or no answer for a given amount of time. When the number disconnects, it will momentarily interrupt your MODEM connection - the very condition you're trying to avoid! So, it may pay to experiment with a few numbers to see which will time out, and which won't. For example, it may be that on your exchange, busy signals will time out but "invalid number" recordings won't. Try dialing a local call with a "1" in front of it, and see if you get a recording telling you not to dial the "1" (or try the opposite-a long distance call with no "1"). That may get you a recording that will not time out.

When you use the Three Way Calling method, there is an additional disadvantage in that if your MODEM call is busy or does not answer, you will have to set everything up again next time you try the call. In contrast, Call Forwarding remains enabled until you specifically disable it.

You may find it advantageous to have BOTH Call Forwarding and Three Way Calling. This way, if someone calls YOU and wants to upload something to you (or you want to download to him), you can first excuse yourself, go onto Three Way Calling, and program your Call Forwarding from there, so that you won't be interrupted. Otherwise, you would have no way to set up Call Forwarding once a call is already in progress.

Of course, by the time you have Call Waiting, Call Forwarding, and Three-Way Calling, you'll be paying a pretty hefty monthly charge to the phone company. So you may find that getting rid of the Call Waiting is an attractive alternative. Or, perhaps you can limit your MODEM calls to after midnight, when no one in their right mind would be calling you anyway (of course, you may know a few folks that aren't in their right mind, or maybe you have a reputation as a night owl, like most computer hackers).

Please note - the above methods of defeating Call Waiting are suggestions only. I assume no responsibility if your phone company gets mad at you for actually using one of them. HOWEVER, if they DO complain, you might suggest to them that if they are going to sell such a service, they might consider providing a way (maybe a one or two digit code) that people can temporarily disable it, so that important calls would not be interrupted. Even if you're NOT using a MODEM, it is an awful nuisance to have a long distance call constantly interrupted by Call Waiting - which is probably one reason the feature isn't more popular than it is!

MAINTENANCE TIPS by Dave Owen is reprinted from Micro Info Exchange:

In my work as a technician, I have encountered a number of Model IIIs whose Drive 0 would not even boot, let alone run the diagnostic software that I normally use to pinpoint problems. I decided to let the Floppy Disk Controller (FDC) tell me about its problems. The solution (printed below) is a BASIC program that can be easily loaded from tape or disk. It is written for the Western Digital 1793 FDC chip with port values used by Radio Shack's Models III and 4. You are free to use and modify this program as you see fit:

```
1 'FDCIII/BAS: WESTERN DIGITAL 1793 FLOPPY DISK
CONTROLLER
2 'TESTER by Dave Owen, c/o CCS, POB 3032, Camarillo, CA,
3 '93011. From "Micro Info Exchange" (Cabrillo Computer
4 'Society Newsletter), August '83, page 5.
5 'If drive 0 on a Model III/4 does not boot, run this test,
6 'It checks the FDC chip--not the disk drives (except to
7 'learn if they are present and active).
8 'If the FDC passes this test, then run Floppy Doctor,
9 'which will extensively exercise the drives themselves
10 CLS:PRINT"TEST 1: Disk Controller Ready Check...Verify
Status Bit 7 set":PRINT
20 OUT 240,208
30 GOSUB 620
40 IF ST=255 THEN PRINT"No power applied to disk controller
chip!"
50 GOSUB 830
60 PRINT"TEST 2: Disk Status Checks...Verify Status Bit 2
set":PRINT
70 PRINT"   ...DRIVE 0: ":B=129:GOSUB 940
80 PRINT"   ...DRIVE 1: ":B=130:GOSUB 940
90 PRINT"   ...DRIVE 2: ":B=132:GOSUB 940
100 PRINT"   ...DRIVE 3: ":B=136:GOSUB 940
110 GOSUB 830
120 CLS:PRINT"TEST 3: NMI Register Checks...Verify no
errors":PRINT
130 IR=INP(228):PRINT"   ...CONTROLLER: ";
140 IF IR=62 THEN PRINT"No interrupt--not ready":GOTO 190
150 IF IR=126 THEN PRINT"No interrupt--ready":GOTO 190
160 IF IR=190 THEN PRINT"Interrupt requested--not
ready":GOTO 190
170 IF IR=254 THEN PRINT"Interrupt requested--ready":GOTO
190
180 PRINT TAB(10)"ERROR--interrupt failure"
190 PRINT"   ...DISK DRIVE 0: ";:OUT 244,129:GOSUB 880
200 PRINT"   ...DISK DRIVE 1: ";:OUT 244,130:GOSUB 880
210 PRINT"   ...DISK DRIVE 2: ";:OUT 244,132:GOSUB 880
220 PRINT"   ...DISK DRIVE 3: ";:OUT 244,136:GOSUB 880
230 GOSUB 830
240 PRINT"TEST 4A: Track Register Checks...Verify no
errors":PRINT
250 FOR A = 0 TO 255
```

```
260 OUT 241,A:B=INP(241):PRINT@132,"...TRACK ";A
270 IF A<>B PRINT@196,"ERROR TRACK ";A
280 NEXT A
290 PRINT@320,"TEST 4B: Sector Register Checks...Verify no
errors":PRINT
300 FOR A=0 TO 255
    OUT 242,A:B=INP(242):PRINT@452,"...SECTOR ";A
    IF A<>B PRINT@516,"ERROR SECTOR ";A
330 NEXT A
340 PRINT@640,"TEST 4C: Data Register Checks...Verify no
errors":PRINT
350 FOR A=0 TO 255
360 OUT 243,A:B=INP(243):PRINT@772,"...DATA = ";A
370 IF A<>B PRINT@836,"ERROR AT DATA VALUE ";A
380 NEXT A
390 GOSUB 830
400 PRINT"TEST 5: Track Seeking Checks...Verify Status Bit 4
NOT set":PRINT
410 PRINT"  ...DRIVE 0: ":B=129:GOSUB 470
420 PRINT"  ...DRIVE 1: ":B=130:GOSUB 470
430 PRINT"  ...DRIVE 2: ":B=132:GOSUB 470
440 PRINT"  ...DRIVE 3: ":B=136:GOSUB 470
450 GOSUB 830
460 GOTO 10
470 OUT 240,B
480 OUT 244,3
490 FOR X=1 TO 500
500 OUT 244,B
510 NEXT X
520 OUT 244,B
530 OUT 241,0
540 OUT 243,39
550 OUT 240,23
560 FOR X=1 TO 500
570 OUT 244,B
580 NEXT X
590 OUT 244,B
600 GOSUB 620
610 RETURN
620 DN=INP(240):ST=DN
    IF DN-128>=0 THEN DN=DN-128:D7=1 ELSE D7=0
    IF DN-64>=0 THEN DN=DN-64:D6=1 ELSE D6=0
650 IF DN-32>=0 THEN DN=DN-32:D5=1 ELSE D5=0
660 IF DN-16>=0 THEN DN=DN-16:D4=1 ELSE D4=0
670 IF DN-8>=0 THEN DN=DN-8:D3=1 ELSE D3=0
680 IF DN-4>=0 THEN DN=DN-4:D2=1 ELSE D2=0
690 IF DN-2>=0 THEN DN=DN-2:D1=1 ELSE D1=0
700 IF DN-1>=0 THEN DN=DN-1:D0=1 ELSE D0=0
710 PRINT TAB(32)"S7 S6 S5 S4 S3 S2 S1 S0"
720 PRINT TAB(10)"FDC STATUS REGISTER
="D7D6D5D4D3D2D1D0
730 PRINT" = "ST
740 IF D7=1 THEN PRINT TAB(10)"...Bit 7 set: Drive motor not
up to speed":RETURN
750 IF D6=1 THEN PRINT TAB(10)"...Bit 6 set: Diskette write
protected"
760 IF D5=1 THEN PRINT TAB(10)"...Bit 5 set: Disk drive door
open/No diskette
770 IF D4=1 THEN PRINT TAB(10)"...Bit 4 set: Seek error - track
not verified"
780 IF D3=1 THEN PRINT TAB(10)"...Bit 3 set: CRC error in ID
field"
790 IF D2=1 THEN PRINT TAB(10)"...Bit 2 set: Disk drive at
Track 00"
800 IF D1=1 THEN PRINT TAB(10)"...Bit 1 set: Index mark
detected"
810 IF D0=1 THEN PRINT TAB(10)"...Bit 0 set: No disk/ette in
system"
820 RETURN
830 PRINT
840 PRINT"*** Press any key to continue ***"
850 A$=INKEY$:IF A$="" THEN 850
860 PRINT:CLS
870 RETURN
    FOR X=1 TO 500:NEXT X
    IR=INP(228)
900 IF IR=126 THEN PRINT"Ready flip-flop set":GOTO 930
910 IF IR=254 THEN PRINT"Interrupt and ready flip-flop
set":GOTO 930
920 PRINT TAB(10)"ERROR: A problem exists--check further"
930 RETURN
940 OUT 244,B
950 FOR X=1 TO 500
960 OUT 244,B
970 NEXT X
980 OUT 244,B
990 OUT 240,3
1000 FOR X=1 TO 500
1010 OUT 244,B
1020 NEXT X
1030 GOSUB 620
1040 RETURN
```

QUICK/CHEAP COMPUTER TABLE by John Roy is reprinted from the Voice of the '80 Newsletter:

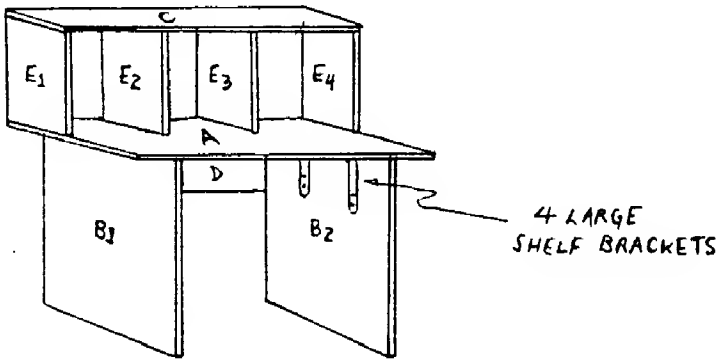
As it is usually the case, the shoemaker goes without shoes, and indeed I still do not have an official computer table. Let me try to make a little more sense. I've had my TRS-80 Model I for about four years and still have it on a temporary card table, with my printer sharing the living room end table. God bless my wife for putting up with this condition for so long. Well, now you probably have guessed that I finally did something about it before divorce court, right? ... Wrong!

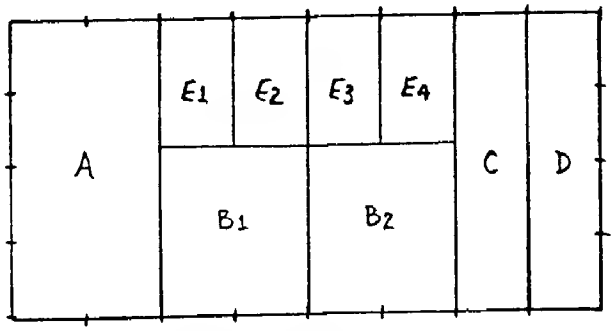
Let me get to the point. My son invested in a TRS-80 Model III and brought it to college with him this fall. Starting to see the picture now? Since he needed a table for his room, the shoemaker came to the rescue. What resulted was the QUICK/CHEAP COMPUTER TABLE.

The object of this table was to use a single sheet of material with some simple cuts. Particle board was chosen for his table, with a cost under ten dollars. Four large shelf brackets, some glue, screws, and nails were the only other items required unless paint or stain is desired. The sketch should show very clearly just how to make the cuts. Dimensions can be altered easily to customize for specific needs. Some key modification areas are the shelf heights and quantity of spacers shown as (E1) through (E4). The shelf width (C) can be widened and the remaining rear support (D) piece is not critical. What is not shown, but was added to his table, was feet to the base of (B1) and (B2). This was accomplished by grooving a piece of hardwood and gluing. Another suggestion, to strengthen the front, is a support similar to (D). This should only be necessary if the material or thickness selected flexes too much.

The project can be completed in a few hours, short any painting or staining. As for the shoemaker, he has decided the wife's patience is nearing its end and will be building his own table.

[Voice of the '80 Editor's Note: The shoemaker's wife reports, and I have seen first hand, the finished product cut from plywood veneer. Hi-gloss finish applied to it provides a place of honor for that dinosaur. I think perhaps that cabinet making, rather than engineering, might be an underlying vocational trait here.]





4x8x 1/2 or 3/4
MATERIAL OPTIONAL

DANGEROUS BENDS - Once in a while, I read about some use for the computer that chills my blood a bit. Longtime readers of this publication know that I have long contended that the computer is a tool, which can be properly used or misused - somewhat like atomic energy. Some examples of computer misuse (such as pornographic video games) are fairly obvious, but others are more subtle.

Such, I think, is the case with a suggestion that appeared in an exchange newsletter that we receive. This publication suggests that hypnosis, or "suggestive therapeutics" as it is called by some, can be used for behavior modification, anestesia (for medical or dental operations), and so on. It states that "the problem with hypnosis is that it requires practice and time", and that because of this, many professionals that attempt hypnosis may have problems and will therefore return to more conventional forms of treatment (anesthetics or drugs).

But help is on the way, suggests this article, in the form of the friendly computer. "There are now programs that will use the visual effects of high resolution graphics and sound to induce hypnotic states. Since most people respond to suggestion when visual patterns match their alpha brain wave rhythm frequency (12-16 cps), selecting the correct frequency, video design, color and sound can induce a patient for glove anestesia levels in 90 seconds. Once you have the correct parameters for each patient, they can be preset for rapid induction at each subsequent visit." This article is, of course, mostly concerned with medical/dental uses, but goes on to suggest any traditional use of hypnosis could be aided by the computer.

The problem here is that there are some folks that should not be hypnotized. Actually, in the opinion of some Christians, no one should allow themselves to be hypnotized! This is probably one of the big differences between Christianity and the eastern religions, whose devotees spend a lot of time trying to put themselves into trances, but my point here is that some folks may have religious objections to hypnosis. However, religious considerations aside, if a person is mentally unstable or schizophrenic, going under hypnosis could be the "trigger" that pushes that person "over the edge".

In the past, a practitioner of hypnosis would normally be expected to have a certain level of competence, such that he would not attempt to hypnotize an unstable person. But, if a computer program that can be used by anyone is available to aid in hypnosis, it means that any unskilled (or worse) person could use the program. Even if the intent is good, we do not fully understand the effects of hypnosis, and do not know what side effects might develop. Further, the doctor (or whomever) would have a perfect opportunity to plant suggestions that might be somewhat at odds with what the patient would do without benefit of the "planted " suggestion (this could be anything from "you will see your dentist every six months" or "you will pay your bill on time" to possibly more suggestive suggestions, particularly if the patient is of the opposite sex). I'm not saying that most users WOULD do this, just that the program conceivably COULD be used in this way.

With such a program available (and at least one already is), a person could even hypnotize himself. You've heard the old saying about a person that represents himself in a court of law, having a fool for a client? Well, an unstable person using this program would be under the care of an unstable hypnotist.

And, of course, a program to hypnotize could be packaged as part of another program. I wouldn't be a bit surprised to find that some video games have "hypnotizing" qualities, they way some folks get "hooked" on them. But suppose that someone wrote a video game with the specific intent of hypnotizing the user as he plays, then planted a "subliminal" suggestion (perhaps some text flashed at various places on the screen, for only a fraction of a second) - perhaps to "play another game" (or something worse!).

The thought that computers could be used to hypnotize occured to me a long time ago, but I never mentioned it because I didn't want anyone to do it. But now that someone has anyway (and even though in this particular case I am sure that the author's intentions are good), I feel that a warning needs to be sounded. Hypnosis is dangerous, we don't know that much about it, and while some folks appear to suffer no ill effects (and even then, we can't be 100% sure), there are others who should NOT be hypnotized, especially by a non-professional.

A few of you may be tempted to write a hypnosis type program anyway. Well, I'd advise that you forget it, since you really don't know what you're messing with. And if that isn't enough to dissuade you, keep in mind that if someone else uses your program and then goes out and commits some violent crime (against himself or someone else), you could wind up facing civil or even criminal legal action against you (stranger things have happened, especially in this country where some folks seem to be "lawsuit happy").

Note that I never said that hypnosis doesn't work. The problem is that it DOES work - but you can never be sure beforehand exactly HOW it's going to work, or what the after-effects will be. So, don't try to hypnotize anyone, don't allow yourself to be hypnotized, and watch out for computer programs that flash at a rate of 12-16 cps!

NEWDOS/80 VERSION 2 ZAPS (for the Model III version of NEWDOS/80, unless otherwise specified) - provided by Tony Domigan:

1. To boot in lower case:

SYS0/SYS,11,2F change 28 05 to 00 00

2. To boot in 4Mhz on Model 4:

SYS0/SYS,13,B4 change 00 00 00 00 00 00 00 00 00 00 00 00 00
to 3A 10 42 F6 40 32 10 42 C3 B6 48 00
SYS0/SYS,11,DF change C3 B6 48 to C3 A8 50
Also, execute the following SYSTEM command: SYSTEM,0,BJ=2

3. When running on the Model 4 in the Model 3 mode, NEWDOS/80 version 2.0 does not get the date and time correct on warm boot because it uses the longer heartbeat counter applicable to the Model III. To correct date and time updating on warm boot:

SYS0/SYS,02,20 change 1E 20 0D 36 1E 21 to 19 20 0D 36 19 21

4. For users who prefer the DD/MM/YY format to MM/DD/YY - Patch Date prompt on boot and reset (if active):

SYS0/SYS,12,2B change 20 ED to 00 00
SYS0/SYS,12,3E change 20 ED to 00 00
SYS0/SYS,13,4E change 4D 4D 2F 44 44 2F to 44 44 2F 4D 4D 2F
(NOTE: The following patches will accomplish the same thing on the MODEL I ONLY:)
SYS0/SYS,12,67 change 20 ED to 00 00
SYS0/SYS,12,7A change 20 ED to 00 00
SYS0/SYS,13,87 change 4D 4D 2F 44 44 2F to 44 44 2F 4D 4D 2F

5. To allow DD/MM/YY in FORMAT command (Models I & III):

SYS6/SYS,12,45 change 32 to 34

MODEL III SCRIPSIT PATCH by Edward A. Pearson - I have a patch for Model III Disk SCRIPSIT that might be of interest. This patch allows it to read 1500-baud cassette tapes. There might be others out there who converted tape SCRIPSIT to work at 1500 baud, only to find out, later, that Disk SCRIPSIT won't read their tapes. The patch is:

PATCH SCRIPSIT/CMD (ADD=5222,FIND=AF321142,CHG=321142AF)

This will write 0C9H into 4211H on initialization, rather than zero, enabling 1500-baud cassette input.

NEWDOS/80 ZAP by Lyman Epp (comments by Nathan W. Harrington) - This is an optional zap to the Model III version of NEWDOS/80, that will add SYSTEM flag CB to say whether you are operating on a Model 4 computer. This will turn on the high speed clock whenever you boot up:

```
SYS0/SYS,11,12:  change  45 3A A5 43 B7
                 to      45 CD A8 50 B7
SYS0/SYS,13,B4:  change  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                 to      3A FF 43 CB 7F 28 05 21 10 42 CB F6 3A A5 43 C9
SYS17/SYS,02,57: change  FE FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
                 to      FE FE FE FE FE FE FE FE FE FE FE FE FE FE 7F FF
```

If you want this to be done automatically (without a SYSTEM command), then the following zap will do the trick. This will work on Model III NEWDOS/80 operating on a Model III or 4 computer (this will not hurt on a Model III because bit 6 at memory location 4210H is unused. I made this a SYSTEM command because in some programs the high speed clock is unwanted):

```
SYS0/SYS,11,12:  change  45 3A A5 43 B7
                 to      45 CD A8 50 B7
SYS0/SYS,13,B4:  change  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                 to      21 10 42 CB F6 3A A5 43 C9
```

NOTE: You will need to set SYSTEM option BJ=2 for disk I/O to work correctly. You may also want to set option AV equal to twice what you have it under the normal operating speed.

USE YOUR TRS-80 TO STUDY THE SCRIPTURES - If you are a student of the Bible, or especially if you're in any kind of ministry of the Bible, you'll appreciate a program by Scripture Software. The program is called "BIBLE SEARCH" and it permits you to search the entire New Testament (King James Version only) for any word or portion of a word (more than one word or sequence of characters may be searched for at the same time). In this way, the program operates like a computerized Concordance. It will find all verses containing one or more of the given search strings, and display a list of those verses. References can be added or deleted manually, and the entire reference list can be saved to or retrieved from a disk file. There's also a function to sort references into Biblical order.

The user can call up any of the referenced verses, and examine it alone or in context (with surrounding verses also displayed). You may "flag" (or "unflag") any verse thus displayed, and later mass delete all flagged or all unflagged verse references. The text for all referenced verses may be loaded into memory and displayed on the video, or printed out. You can even print multiple copies of the referenced verses for use in a group Bible study.

The program and data files (which contain the King James Version of the New Testament) are available in both a Model I version (supplied on single density diskettes) and a Model III version (supplied on TRSDOS 1.3 double density format diskettes). The program itself will work under any Model I or Model III operating system. The price is $140 and the Old Testament database is also available at extra charge.

The only problem with this system is that in order to search the entire New Testament (or entire Bible) for a given subject, you will have to swap diskettes several times. However, both the program and the data files are in standard (copyable) format, so they can be backed up to a higher-density diskette. The entire New Testament and the BIBLE SEARCH program itself will fit on one 80 track double sided double density diskette, should you be lucky enough to have that kind of equipment. Obviously, there's no way to shorten the Bible (not without coming under the curse of Revelation 22:19, anyway) so if you're a serious Bible student you may want to invest in at least one high capacity disk drive (or perhaps practice patience while swapping disks...).

Beyond that, I personally feel that the price may be a bit steep, especially considering that the data files are the King James Version of the Bible, which is in the public domain. I would personally prefer to see a modern-English version of the Bible used - my favorite is the New International Version - but unfortunately, that version (and almost every other modern-English version) is copyrighted, which might restrict the use that could be made of the program if that version were used. I personally find this a deplorable situation, that the only version of the Bible that can be freely used today (without fear of violating someone's copyright) is a 400 year old version written in antiquated English. I don't mean to step on anyone's toes, but I've always had considerable difficulty with "Olde English" (I couldn't understand Shakespeare when it was forced on me in High School, either). Note that when the Bible is translated into a new language for some tribe in Africa, we don't write it in a 400 year old version of their dialect, so why should the English-speaking majority of the world's population have to make do with the old version? (even though it was an excellent translation in its day - unfortunately, the English language has changed a lot since then!). For the record, I don't care for paraphrased editions or certain other modern "translations" that are partly wishful thinking on the part of the author(s), so you don't have to write and tell me about the dangers inherent in those editions - I'm fully aware of them.

Anyway, next time you get into a discussion about the copyright laws, keep in mind that they are partly responsible for the fact that most English-speaking people still use the King James version of the Bible, rather than a more up-to-date translation, for quotations and other general use. Whether you find that a blessing or a curse will depend upon your point of view about the KJV. End of editorial.

If you are comfortable with the King James version, and can afford the $140 for the program and New Testament data files, you'll find BIBLE SEARCH a great help with your Bible study, whether for your own use, for a use in home Bible study group, or for preparing a sermon or Bible-based lecture. BIBLE SEARCH requires 48K of memory and two disk drives (that's if you're using standard drives where everything won't fit on one disk), and is available from Scripture Software, P.O. Box 6131-C, Orlando, Florida 32853. For further information (or if you have a different make or model of computer and would like to inquire if a version of BIBLE SEARCH is available for your computer), you may also telephone Scripture Software at (305) 896-4264.

THE FIRST CHECKSUM (REVISITED) - I have received a couple of letters in regard to this article (which discussed Ivan Panin's work in Bible numerics, and appeared in NORTHERN BYTES, Volume 5, Number 2). Rather than reprint the correspondence (which would take up quite a bit of space), let me summarize it here.

Dr. Michael W. Ecker (129 Carol Drive; Clarks Summit, Pennsylvania 18411) is a mathematics professor with a PhD in that subject, and a self-described "atheist by conviction." Dr. Ecker took issue with my coverage and felt that the word "proof" had been used indiscriminately within the article. He also raised the issue of "selective perception." Here's an excerpt from his letter:

"...the fabulous odds you quote require careful examination. I can, for instance, obtain a poker hand with the 2 and 3 of clubs, the 6 of hearts, the 8 of spades, and the queen of diamonds. This is quite a worthless hand. And yet, the odds against obtaining it are astronomical! Yet I did get this hand.

"What I am getting at is the fallacy of selective perception. This is also known as seeing what you want to see. If you look for the number 6, for instance, suddenly your world is filled with sixes! Six faces on a cube, 6 is the first example in mathematics of what we call a perfect number, it is the product of the first two primes, it is equal to both the sum of the first three numbers 1,2,3 as well as the product of them, etc. etc. I am not trying to be that convincing, only to illustrate the point. In a recent issue of the Pennsylvania State Mathematical Association of Two-Year Colleges newsletter which just came out (I will send a copy to you or anybody who requests it), by an interesting coincidence there was a piece on this fallacy and teaching students to understand and avoid it. The example there used the improbable number 47, and suddenly everyone was finding the number 47 all over the place.

"In order for something of this sort to have any validity, for starters you would have to take somebody who had no idea as to what he was looking for and ask him! If you consider all different things you can think of doing with the passages in this book, which number strikes you as dominant or recurring? - and even pick an individual who is mathematically inclined. Only if he and others pick the same number (7 in the article you printed) consistently would further consideration be warranted. Any number can be found very often if you keep looking!"

Although Dr. Ecker does not indicate in his letter that he has ever done any actual research into Panin's work (would would put him in a much better position to criticize the article), he does bring up an interesting point. I can understand what he is talking about because even in Christian circles, there are people who find certain numbers everywhere. For example, I know people who see the number "666" everywhere (see Revelation 13:18 for the significance of this number), and while I agree that it is certainly worth noting when it pops up in something like the start/stop bit patterns of the Universal Product Code symbols used on supermarket products, I hardly find anything sinister when it turns up in a normal rotation - for example, if three digit numbers are used on the automobile license plate numbers in your state or province, it stands to reason that one out of every thousand persons is going to get a "666" - and in a state capital, there's even a reasonable chance that a government official could draw the number. In other words, having "666" on your license plate proves nothing - but if, for example, many high government officials came up with "666" on their license plates, and relatively few non-government folks got that number, then we might have something to be concerned about. In other words, finding a frequent occurrence of a number proves nothing unless that number occurs considerably more frequently than other numbers (all other circumstances being equal).

I think that the point of Panin's research was that the number seven (and derivatives of the number 7) occurred much more frequently in the Bible than might be expected in normal Greek or Hebrew text. But I do not have the time to research this further, in order to prove or disprove it. That brings me to the other letter I received, from John Easton of Christian Computer Based Communications (44 Delma Drive; Toronto, Ontario, Canada M8W 4N6). Here's an excerpt from John's letter:

"[My] real reason for writing was in response to your mention ... of Panin's work on Greek & Hebrew numerics. We have more stuff on file (including a copy of Panin's book on the subject

- strictly for the scholarly types), which might be of interest to your readers should they like to contact us here. We have a small photocopied booklet on this phenomenon, and as an aid to checking out the claims made by Panin, a program written for the Commodore '64 (because of it's programmable character capabilities) of a full Greek/Hebrew Numerics Report Generator. I'll enclose a printout of the listing for your perusal. Generally, if one pays no attention to things like POKE (usually associated with screen colour and/or character definition) this should translate in a fairly straightforward manner to other machines. It is meant to function with a specific word processing package, but text processing shouldn't prove insurmountable. And I seem to recall there is a Greek/Hebrew Character file on the disk for loading into an EPSON FX-80 for printout."

There you have it - if you are interested in this subject and would like to research it further, why not drop a line to John and ask him for more information and/or a copy of the program listing (which runs about 4-1/2 pages). It costs 37 cents to mail a letter from Canada to the U.S. and they cannot use U.S. postage stamps, so if you're requesting information, you might send about 40 cents in coin to help defray expenses (U.S. coins spend perfectly well in Canada!).

[Editor's plea - If you want to discuss this subject further, it would be highly appreciated if you could send your comments on TRS-80 compatible media (disk or tape - I'll even take text embedded in BASIC remark statements if necessary!), or better yet, send it to me electronically via MCI Mail! I positively hate to retype text into the computer!]

LETTER TO THE EDITOR

Dear Jack,

Two questions I would like to see discussed in Northern Bytes is:

A. How to use key files to access large data files i.e. 2000 records or more. My keyfile (2000, 8 byte keys) is loaded into memory. The continuous string manipulation occasionally gives me a garbage collection. Using MID$ and POKEing into 15669 helps but sometimes causes problems.

B. How can I use the 2 upper 32K bytes in a Model 4 128K computer to hold a key file? I have looked through DOSPLUS IV, CP/M 3.0, TRSDOS 6.01 and the best I can find is: Use it with GETs and PUTs to store info on a fast disk. Which means it simply saves time but does not effectively increase memory.

(signed) Bro. J. Smith, St. John's High School, Shrewsbury, Massachusetts 01545

[Editor's Reply - You have to know exactly what is going on within a program if you want to eliminate "garbage collections" using the methods you've tried. Some BASIC statements and functions create temporary strings, and if you don't realize that this is happening it may thwart your efforts to delay collections.

Modular Software Associates produces a program called "The Collector" that eliminates or greatly reduces "garbage collection" delays. It's similar to Prosoft's "Trashman" program, but uses less memory and is less expensive (it's sold by The Alternate Source for $24.95 plus the standard $3.00 shipping/handling charge). The only drawback of using a program like this is that it uses two bytes of memory for each active string, so if your program uses nearly all available memory now, this option may not work for you.

The method you are using to access the two additional 32K banks of memory in the Mod 4 is the only practical method that I know of to use that additional memory from BASIC. Unfortunately, both versions of BASIC supplied with the Model 4 (the Model III BASIC in ROM, and the version of MBASIC supplied with TRSDOS 6.x) are designed to directly access only the 64K of memory that is directly addressable by the Z80 microprocessor. It is very possible that in the future, a new version of MBASIC or an enhancement program that modifies the operation of one of the supplied versions of BASIC might appear, that would provide an easier way to utilize the additional 64K of "add-on" memory.

I might add that at this writing, TAS is aware of a program that is functionally similar to the MEMDISK program of TRSDOS 6.x, but that works with NEWDOS/80 in the Model III mode. If we are able to acquire the right to market this program, you'll see an announcement to that effect in the near future!

If any of our readers has a better solution to your specific problem, I hope that they will pass it along to you, and send a copy to us here at NORTHERN BYTES so that we can share it with everyone!]

READ IBM DISKETTES WITH SUPERZAP by William F. Terrell is reprinted from LLIST Newsletter (publication of The Rochester S-80 Computer Club, Inc.):

Have you tried to read IBM PC diskettes with SUPERZAP? Then you've noticed that your TRS-80 goes berserk after reading one sector (if you're lucky). If you make some minor patches to a copy of SUPERZAP and fix your PDRIVE specifications, you'll be able to use it in conjunction with DEBUG to access IBM PC diskettes. Here's how!

First, why the difficulty in reading IBM PC diskettes? It is because the PC uses 512 byte sectors and SUPERZAP has provided only a 256 byte buffer. When you read in a sector from a PC diskette, the last 256 bytes overwrites part of SUPERZAP. Hence the crash!

Our solution is to move the SUPERZAP buffer (presently at 5300H) to a new location which has the following 256 bytes unused. I chose location 8000H. To make a copy of SUPERZAP with the relocated buffer, use SUPERZAP to change every occurrence of bytes 00 53 to bytes 00 80 in this copy.

Next, set your PDRIVE specifications for the drive you will be using for the PC diskette. Use SPT=9 to set for the 9 sectors per track of PCDOS 2.0. Or, use SPT=18 if you have a double sided drive. Obviously, you will not be able to read the back side of the PC diskette without a double sided drive. Set the 'I' flag with the type of interface. I'm using the Radio Shack Doubler on a Model I, so I set TI=DI. You're ready to go!

Use SUPERZAP in the usual fashion. Note that the first track is zero and the first sector is zero. As you page through the PC diskette with the plus and minus keys, you will see the first 256 bytes in each sector. To see the last 256 bytes in a sector, use the 123 keys to invoke DEBUG. Set the DEBUG display for 8100H with the 'B' full screen display. Hit <G><ENTER><ENTER> to return to SUPERZAP's display of the first 256 bytes. Don't expect the DFS function to work since the PC directory is different than TRSDOS'.

While I haven't tried it yet, you should be able to patch an IBM PC file sector as follows. First, modify the last 256 bytes of the sector while in DEBUG. Next, modify the first 256 bytes while in SUPERZAP. The patch can be written to disk in the usual manner using the SUPERZAP MOD command whether or not you actually made a change to the first 256 bytes.

[NORTHERN BYTES EDITOR'S NOTE: The above article is reprinted mainly to prove that at least one person has succeeded (at least partially) in reading an IBM diskette using a lowly Model I TRS-80! In other words, the hardware is willing, but the available software is weak. What we need now is a good utility program that will read and write IBM disks - perhaps to copy to/from a standard TRS-80 disks, or perhaps to add this capability to one or more of the popular TRS-80 operating systems. I have a feeling that a well-written program of this type might be a hot seller, even if its only practical application would be to transfer ASCII files (word processor text files, BASIC programs saved in ASCII format, etc.) between the IBM PC and the TRS-80 Models I/III/4. If you develop anything along these lines, please drop us a line here at NORTHERN BYTES and let us know what you've accomplished!]

CONVERT YOUR CENTRONICS 737 PRINTER TO A CENTRONICS 739 (REVISITED) by Jack Decker - A couple of issues ago (back in NORTHERN BYTES, Volume 5, Number 2) I told you how to upgrade your Centronics 737 line printer to a Centronics 739. The one thing that I couldn't tell you was how to program the 2716 EPROM used in the conversion. Well, thanks to Joachim Kelterbaum, I now have the code you need to put into the EPROM in order to complete the conversion. If you want to make the conversion and can't get the proper EPROM from Centronics, send me a blank disk (or SASE for hardcopy) and return postage and perhaps I can help.

MODEL III DISK BASIC BUG? - According to an article by Don Hallden in the March, 1984 issue of TRS-80 Microcomputer News, a numeric variable cannot be used to define field lengths in FIELD statements. For example, this line would generate a syntax error!

```
L=20: FIELD 1, L AS A$
```

However, according to Don, the VAL function can be used to achieve the same effect. This line WOULD work!

```
L$="20": FIELD 1, VAL(L$) AS A$
```

So, it is possible to get a field length from a variable, but the question remains, why can't you use a numeric variable? Is this a bug, or a design "feature" of Model III Disk BASIC?

MOONLIGHTING WITH YOUR WORD PROCESSOR - This article was written by Jessie Gunn Stephens and was submitted to NORTHERN BYTES by J. Norman Goode, Publisher of MICRO MOONLIGHTER NEWSLETTER (4121 Buckthorn Court, Lewisville, as 75028 - Telephone (214) 539-1115). COPYRIGHT 1984 by lorman Goode.

If you have a personal computer, a printer, and word processing software, you have the wherewithal to start your own business. You can work part time or full time out of your own home, with low overhead and a consistent demand for your services.

If you're a competent typist, you may have already thought about setting up a word processing service bureau in your home. Thousands are doing so across the land and are finding small businesses and professional people, doctors, lawyers, dentists, and so forth, eager to employ them, either on a contract or a piece work basis.

The demand for such services is not confined to large cities, either. In fact, smaller communities, where technology is slow to arrive, sometimes offer the best ground floor opportunities for starting a word processing service bureau. Competition may well be minimal, and potential clients may respond more favorably to higher fees, since their experience with technology is more limited. Of course the word processing bureau that charges more than the local typist must be ready to justify the cost difference. In most cases, this means the highest quality and fastest turnaround in town.

But the WPS bureau is by no means the only avenue to independent income via your word processor. Many proud entrepreneurs have found success in everything from list brokering to freelance magazine writing --- all made possible by the personal computer. Some entrepreneurs write and publish newsletters, others computer documentation, others various kinds of bibliographies and directories.

Word processing has been a boon to freelance writers, no matter how large or small the market they serve. Manuscript mechanics are a breeze on the computer, as are multiple copies and large scale revisions. As the publishing business moves toward tronically controlled operations, writers working at home with r own word processors will have the competitive edge in a ughly competitive field.

In fact, with the approach of the Information Age, the "cottage industries" predicted in Alvin Toffler's bestseller THE THIRD WAVE are beginning to proliferate. People are using their personal computers to provide specialized writing and publishing services to segments of the population as varied as neighborhood mom and pop proprietors and corporate data processing professionals.

At this point, straight and simple word processing is still the most popular service being offered out of private homes and small offices. And it may be the one way to open the doors of opportunity for you, particularly if your word processing package is fairly sophisticated and you are well versed in its use. And if you're a competent typist.

Your business overhead will include not only your computer and a good quality printer (usually letter quality), but also adequate storage media, a good supply of printwheels or thimbles, carbon film ribbons, high quality paper, and a spelling checker program.

The size and range of such a business is up to you. You can start small, perhaps on a part time basis, and work up to whatever work load you want to handle. You can offer any range of services, from straight typing, charged by the page or hour, to whole newsletters, weekly or monthly bulletins from businesses, churches and clubs, product catalogs, handbooks, pamphlets --- whatever your customers need.

If you have specialized knowledge, for instance if you've worked in a law firm or a dentist's office, seek your first customers among people who will appreciate the fact that you're already familiar with the forms they use and the jargon they write. If they're pleased with your work, be sure to ask them to tell their colleagues about your service. Anytime you have a satisfied customer, give him or her a few of your business cards and out for you.

Before you decide what fees to charge and begin soliciting business, though, there are a number of considerations you'll want to work out for yourself. Some of the types of questions you'll want to decide in advance are:

- Will you bill by the hour, page, or document?

- How many copies will you provide?

- Can you handle forms?

- What will you charge for revisions of finished work?

- Can you index?

- How will you protect the clients' material against fire or theft?

Once you know the answer to these and similar questions about the nature of your business, you can set a fee schedule to cover your time and overhead and then advertise your services. One successful entrepreneur got all the work she could handle in the following way:

> "For a modest fee," she says, "I was able to get eleven lines of advertising displayed on cable television for a week. The local newspaper offered a four dollar ad, of which I took advantage. Then I prepared two sets of business letters --- one for lawyers and courts, one for business people --- and hand delivered samples to lawyers and business people in the community."

Another beginner found that advertising her services in a large city newspaper prompted only a few customers to call. But when she put a notice in the local suburban "shopper's" weekly, she had to turn away business. It was evidently the idea of local service that people responded to. She gets most of her business from doctors, lawyers, and other professionals --- people for whom accuracy and quality count a lot. She provides both, and she charges accordingly, with no complaints from clients.

Straight text and legal forms make up the bulk of this entrepreneur's work. As for you, whether you want to process straight text provided by clients, write and sell your own material, or process information for mailing lists, your word processor can be the key to your own full or part time business. The demand for such services is growing. If you can meet it with a high quality product, you can be your own boss.

* * * * *

Ms. Gunn is a widely published freelance writer. Her start-up guide for home word processing services: NEW PROFITS IN WORD PROCESSING is available for $19.95 plus $2.00 Shipping & Handling from: J. Norman Goode, Publisher; 4121 Buckthorn Court; Lewisville, Texas 75028.

DEFEAT BACKUP PROTECTION ON TRSDOS 6.x DISKS by Nathan W. Harrington - Here is a one-liner that will take a TRSDOS 6 diskette and make it un-backup-protected. So far, I know that PFSfile and PFSreport for the Model 4 have limited backups. I found this out because PFSfile came with our Model 4. Angered by not being able to make unlimited backups, I set out to solve the problem. As it turned out, it took about 5 minutes to locate and fix the problem. The following program contains the basic necessities required. You can change the drive containing the diskette by changing the drive number in the OPEN statement. Operation is very simple. Insert the diskette to be unprotected in drive 0 (or whatever you change it to) and run the program. The result, unlimited backups. The program:

```
10 OPEN "R", 1, "BOOT/SYS.LSIDOS:0" : GET 1, 3 : POKE VARPTR (#1) + 198, 0 : PUT 1, 3 : CLOSE : END
```

TASMON PATCH FOR MODEL 4 (provided by Nate Salsbury) - If you have the Model III version of TASMON, and are using it on the Model 4 and find that the SHIFT-* (screen dump) and S (sum) functions don't work correctly, apply the following patch:

PATCH TASMON/CMD (ADD=79B9,FIND=210531,CHG=212531)

This patch is in Model III TRSDOS 1.3 patch format. If you are using a different DOS, you may have to use a different method to make the patch. What you are doing is changing a LD HL,3105H instruction at 79B9H (points at a keyboard scan table in the Model III) to a LD HL,3125H instruction (points at the printer driver table). Nate points out that while this will fix TASMON, it may not provide complete compatibility if some (U)ser routines have been added that use shifted keyboard characters for input, because the Mod III keyboard table and the printer table differ once you get past the '+' sign.

WANTED TO BUY - A 202 (not 212) MODEM that can be hooked up to a TRS-80 Model I via the RS-232 interface. Contact Dave McGlumphy, 4429 Paula Lane, Chattanooga, Tennessee 37415.

COMPDIR REVISITED - Way back when (last year), we published Dave McGlumphy's COMPDIR program (as slightly modified by ye editor) that was used to compare the directories of two NEWDOS/80 diskettes, and send the similarities or differences to video, printer, or a disk file. The disk file option was included to permit easy creation of ILF or XLF files (as used by NEWDOS/80's COPY command). Trouble was, there was one fly in the ointment - if any filename in the directory had an extension of one or two characters in length, the extension would be padded with spaces, and that's not valid in an ILF/XLF file. Since this program has already been debugged a couple of times previously, I decided to renumber it and reprint it in its entirety, to avoid confusion.

If you now have version 2.2 (which, by the way, may be the version you have if you got it from an early TAS Public Domain Library disk), then do the following to upgrade to version 2.3:

1) Renumber the program starting at line 10, in increments of 10 (do a RENUM 10,10 from BASIC "READY").

2) Change lines 10 (the version number), 60 ("<100"), and 90 and 110 (miscellaneous changes) to be the same as the corresponding lines printed below.

Once you've done this, COPY won't puke on your ILF/XLF files that you've created using COMPDIR. I find that I use this program quite a bit, to compare the directories on my 80-track double-sided disks. You may find it useful as well. Here's the BASIC program listing - note that lines 10, 30, and 40 contain embedded linefeeds (these can be inserted during BASIC line entry through use of the down-arrow key):

```
10 REM  COMPDIR 2.3 -- ORIGINAL PROGRAM BY
     DAVE MCGLUMPHY, 4429 PAULA LN, RED BANK, TN 37415
     PROGRAM MODIFIED BY JACK DECKER
20 CLS: CLEAR 6000: DEFINT A-Z : DIM
N(444),P$(444),A$(8),N$(8): FOR J=1 TO 8: FIELD 1, (J-1)*32 AS
X$, 1 AS A$(J), 4 AS X$, 11 AS N$(J), 16 AS X$: NEXT J: M=216:
S=8: X$="INVISIBLE": GOSUB 180: IF M=208 THEN S=64:
X$="SYSTEM": GOSUB 180
30 N=1: GOSUB 160: N=2: GOSUB 160: PRINT"SORTING ...":
CMD"O",C,P$(1),N(1): CLS: PRINT"DO YOU WANT A LIST OF THE
FILENAMES THAT APPEAR ON:

<0> BOTH DISKS
<1> DISK # 1 BUT NOT DISK # 2
<2> DISK # 2 BUT NOT DISK # 1
<3> EITHER OF THE TWO DISKS BUT NOT BOTH"
40 GOSUB 200: A=VAL(X$): CLS: PRINT "DO YOU WANT THE LIST
OF FILENAMES PRINTED ON:

<0> VIDEO DISPLAY ONLY
<1> PRINTER AND VIDEO DISPLAY
<2> DISK FILE AND VIDEO DISPLAY
<3> PRINTER, DISK FILE, AND VIDEO DISPLAY": GOSUB 200:
B=VAL(X$)
50 IF B>1 LINE INPUT "OUTPUT FILESPEC? ";X$: ON ERROR
GOTO 220: OPEN"O",1,X$: ON ERROR GOTO 0
60 L=PEEK(16598): M=PEEK(16599): IF
(L+M*256)-(PEEK(16544)+PEEK(16545)*256)<100 THEN M=FRE(X$):
GOTO60
70 CLS: IF A=3 PRINT "DISK #",
80 PRINT "PROGRAM NAME": PRINT: FOR J=1 TO C: IF
P$(J)=P$(J+1) THEN J=J+1: IF A=0 THEN 90 ELSE 150 ELSE IF
(N(J) AND A)=0 GOTO 150
90 X$="/"+RIGHT$(P$(J),3)+" ": X$=LEFT$(X$,INSTR(X$," ")-1): IF
X$="/" LET X$=""
100 S=INSTR(P$(J)," "): IF S=0 OR S>9 LET S=9
110 X$=LEFT$(P$(J),S-1)+X$: IF (B AND 2) PRINT #1,X$
120 IF A=3 PRINT N(J),
130 PRINT X$: IF (B AND 1) = 0 THEN 150 ELSE IF A=3 THEN
X$=MID$(STR$(N(J)),2)+" "+X$
140 LPRINTLEFT$(X$+STRING$(15,32),16);
150 POKE 16598,L: POKE 16599,M: NEXT J: END
160 PRINT "WHAT IS THE SOURCE DRIVE NUMBER FOR DISK #"
N" (0-3)? ";: GOSUB 200: PRINT X$: ON ERROR GOTO 230: OPEN
"R",1,"DIR/SYS:"+X$: FOR S=3 TO LOF(1): GET 1,S: FOR J=1 TO 8:
IF (ASC(A$(J)) AND M)=16 THEN C=C+1: P$(C)=N$(J): N(C)=N
170 NEXT: NEXT: PRINT "FINISHED READING ALL" S-3
"SECTORS": CLOSE: ON ERROR GOTO 0: RETURN
180 PRINT "INCLUDE "X$" FILES? ";: X$=INKEY$
190 X$=INKEY$: IF X$<>"N" AND X$<>"n" AND X$<>"Y" AND
X$<>"y" THEN 190 ELSE PRINT X$: IF X$="N" OR X$="n" THEN
RETURN ELSE M=M-S: RETURN
200 X$=INKEY$
210 X$=INKEY$: IF X$<"0" OR X$>"3" THEN 210 ELSE RETURN
220 PRINT "BAD FILESPEC OR DISK I/O ERROR - TRY AGAIN":
RESUME 50
230 IF ERR=114 RESUME NEXT ELSE ON ERROR GOTO 0:
RESUME
```

CALLING MCI MAIL FROM CANADA - Since we published the article on MCI Mail in NORTHERN BYTES Volume 5, Number 2, I've had a couple of questions regarding accessing MCI Mail from Canada. First of all, it should be noted that MCI Mail can be used to access Dow Jones News/Retrieval, but the reverse is also true - Dow Jones News/Retrieval can be used as a "gateway" into MCI Mail. The difference is that when Dow Jones is used as the "gateway", a per-minute charge applies while accessing MCI Mail directly incurs NO per-minute charge (U.S. users that are currently accessing MCI Mail through Dow Jones would do well to get their own "direct access" MCI Mail account, so that they can access MCI Mail directly without paying the per-minute charge). There may be other services that offer a "gateway" into MCI Mail, however, almost any of these will incur a per-minute charge when used, plus (possibly) an additional Bell Datapac surcharge when accessed through that system in Canada.

MCI Mail has indicated that at some point they may extend service to Canada, possibly by installing an (800) INWATS service line that could be accessed from Canada. If you would like this service, contact them and let them know about it (write Marketing Department, MCI Mail, 2000 M Street N.W., Washington, D.C. 20036). In the meantime, if your business could stand to profit by being hooked up with MCI Mail, you might consider using one of the U.S. local access numbers (if you can call between midnight and 8 A.M., the rate would be relatively low, especially if a 1200 baud MODEM is used). Here's a few suggestions! In the Atlantic provinces and Quebec, try Boston at (617) 262-6468. In Ontario, nearby points are Buffalo at (716) 847-6050, Rochester at (716) 955-9850, and Detroit at (313) 962-5980. Users in central Canada can try Milwaukee at (414) 347-1769 or Minneapolis at (612) 893-9462. Western Canadians will probably have to call Sacramento at (916) 442-6986 or Oakland at (415) 540-1114 (believe it or not, there is no local number listed for Seattle as of this writing!). Keep in mind that you can register for MCI Mail by MODEM, type the word "REGISTER" for both User I.D. and password and follow the instructions (see the article in Volume 5, Number 2 for more information). Once you're an MCI Mail user, you can type HELP PHONES every so often, and possibly find a new phone access number nearer your location.

A ZERO-COST HARDWARE PERIPHERAL by Alex Auerbach is reprinted from the Valley Computer Club Newsletter:

Having taken on the job of printing the club newsletter mailing labels, I was faced with a problem. I have an Okidata printer with pinfeeds on the platen, but without a tractor. I am too cheap to buy a tractor, and didn't want to buy labels on 9 1/2 inch wide pinfeed carrier paper, since they cost much more per label than the standard one-wide mailing labels. But using the pinfeeds on only one end of the platen caused the labels to shift in the printer.

My solution is elegantly inelegant, but appropriately cheap. I made a tunnel of cardboard, as wide (exactly) as the label carrier paper, about a quarter of an inch high, and a couple of inches deep. I then taped it atop the printer, and ran the labels through it, down around the pinfeeders on one end of the platen, and out. The tunnel (a flattened loop of wire might have worked as well, I realized after my fingers were covered with glue) guides the labels quite nicely without any shifting, and the pins on the one side do a fine job of pulling the carrier through. Now, if I make it in Hong Kong and advertise it in Byte...

FOR SALE - TC-8 High Speed Cassette Unit ("Poor Man's Floppy") with homebrew printer interface installed. Includes TC-8 operating system software. Also includes attached printer cable with edge card connector for Centronics 737 or 739 (Line Printer IV) parallel printer (connector could be changed to standard parallel printer connector to work with other brands of printers). Sold as is, on a "worked when removed from service" basis (was replaced by a disk system). Great for the kid in your computer club that bought an old Model I, and wants to run a printer (or hates the 500 baud cassette speed!) but can't afford an expansion interface. Will sell for first $100 or best offer of $50 or more. Contact editor of this publication (see return address on back page). P.S. Need a printer as well? Make me an offer on the above unit plus my Centronics 739 (upgraded from 737, now has graphics capability) letter-quality printer (same printer used to print NORTHERN BYTES!).

Note! - The Folks that brought you Tasmon and Trackcess

THE

A-L-T-E-R-N-A-T-E

AN OFTENTIMES SUPERIOR SUBSTITUTE

SOURCE

WHERE ORIGINAL NEWS CAN BE OBTAINED

PROUDLY PRESENTS EDX

FULL SCREEN TEXT EDITOR

* Learns your keystrokes
* What you see is what you get
* Create an in-memory image of your pages
* Twenty-six in-memory text storage buffers
* Automatic column formatting for tabular text
* Page editing mode for files larger than memory

SATISFIED EDX USERS WRITE US !

. . . . . EDX has been a real pleasure . . . . . very powerful, practical, and reasonably priced . . . . . EDX should join the hall of fame along with Tasmon and Trackcess . . . . . Most versatile program available . . . . . have experienced no problems with EDX . . . . .

Winston Atkinson Columbia, MO

. . . . . A very good value at the stated list price . . . . . Overcomes the limitations of TRS 80 hardware as well as any editer can . . . . .

David Webster Houston, TX

TRS-80 MODEL I and III.....$29.95

THE ALTERNATE SOURCE 704 N. PENNSYLVANIA LANSING MI 48906

SEND ME
__ Copy(s) EDX Editer....@ $29.95 _____
__ Copy(s) CHEKBOOK 4.0..@ $49.95 _____
__ Copy(s) ABASIC.....@ $69.95 _____
__ Copy(s) ADVENTURE SYS.@ $39.95 _____
Total = _____
Add for Shipping and Handling + 3.00
Amount Enclosed = _____

MY ADDRESS IS

_____
_____
_____
_____

I use a TRS-80 MODEL _____ ?

We also accept Visa, Mastercharge, and COD Orders - Phone 517-482-8270

ALSO NEW FROM

THE ALTERNATE SOURCE

ABASIC 69.95

* Moduler programming in BASIC
* Edit with your word processor
* No line numbers neccessary
* Create libraries of relocatable code
* Support structured constructs
* Label Routines and Subroutines

CHEKBOOK 49.95

* Deducts Service charges
* Provides detailed reports
* Adds Interest income
* 420 user definable catagories
* Deducts Automatic payments
* Supports over 1220 entries

THE ADVENTURE SYSTEM 39.95

* You get five free adventures written by past customers
* Make heroes and monsters out of the people you know best
* Complete high level adventure language in machine Language

You may use the order blank on the other side of this page

NORTHERN BYTES
c/o Jack Decker
1804 West 18th Street
Lot # 155
Sault Ste. Marie, Michigan 49783
MCI Mail Address: 102-7413
Telex: 6501027413
(Answerback: 6501027413 MCI)

Bulk Mail
U.S. Postage Paid
Permit 815
Lansing, MI

POSTMASTER: If undeliverable return to:
The Alternate Source, 704 N. Pennsylvania, Lansing, MI 48906

To: